UNIVERSIDAD POLITECNICA DE MADRID
FACULTAD DE INFORMATICA

TRATAMIENTO N—CATEGORIAL DE LA LOGICA DE LA PROGRAMACION

TESIS DOCTORAL

José ARTECHE OTEGUI
Ingeniero de Telecomunicación.

1 9 8 9

UNIVERSIDAD POLITECNICA DE MADRID - BIBLIOTECA
FACULTAD DE INFORMATICA

1000193955
T. 58
R. 58

DEPARTAMENTO DE METODOS ESTADISTICOS I.O. E INTEL. ARTIFICIAL

FACULTAD DE INFORMATICA DE LA U.P. DE MADRID

TRATAMIENTO N-CATEGORIAL DE LA LOGICA DE LA PROGRAMACION

Memoria presentada por

José ARTECHE OTEGUI, Ingeniero de Telecomunicacion,

para la obtención del grado de Doctor en Informática.

Director, Prof. D. Luis de LEDESMA OTAMENDI

Profesor Titular de Ciencias de la Computación e Inteligencia Artificial, de la U.P. de Madrid.

Codirector, Prof. D. Luis Maria LAITA DE LA RICA

Catedrático de Ciencias de la Computación e Inteligencia Artificial, de la U.P. de Madrid.

Mayo de 1989

TRIBUNAL

encargado de juzgar la Tesis Doctoral:

Presidente: Prof. D. Sixto Rios Insua
Catedratico de la U.P. de Madrid.

Vocales: Prof. D. Enric Trillas Ruiz
Catedrático de la U.P. de Cataluña.
Prof. D. Julio Gutierrez Rios
Catedrático de la U.P. de Madrid.
Prof. D. Eladio Dominguez Murillo
Profesor Titular de la U. de Zaragoza.

Secretario: Prof. D. José Cuena Bartolomé
Profesor Titular de la U.P de Madrid.

Suplentes: Prof. D. José Fernández Prida
Catedrático de la U. Complutense.
Prof. Dña. Ana María García Serrano
Profesor Titular de la U.P. de Madrid.

TRATAMIENTO N-CATEGORIAL DE LA LOGICA DE LA PROGRAMACION

En el contexto de las técnicas de validación de programas, de las estructuras de axiomatización de la programación planteadas por Hoare y del desarrollo de la Lógica de la Programación que Dijkstra realiza con el operador wp, hemos dotado a los elementos esenciales que intervienen en dicho desarrollo de una estructura que permite su estudio en el ámbito de las categorías.

En este trabajo se demuestra que el conjunto de precondiciones de un fragmento de programa anotado tiene estructura de N-categoría y que sucede lo mismo con el conjunto de postcondiciones. Más aún, se ha puesto de manifiesto (y probado) que el operador wp actúa como un funtor entre estos pares de N-categorías.

Además, los conjuntos de guardas surgen de un modo natural en la lógica de la programación, han sido tratados desde el punto de vista de la semántica denotacional (Wirth para el PASCAL y Scott y otros, después, en un enfoque más general) y presentados por Manes y Arbib en su semántica parcialmente aditiva. Pues bien, en este trabajo se demuestra que los conjuntos de guardas tienen

también estructura de N-categoría y que cualquier N-categoría dotada de una suma definida adecuadamente tiene estructura de conjunto de guardas, de tal modo, además, que el preorden inducido por la suma en el conjunto de guardas coincide con la flecha de la N-categoría.

Esta suma es, en concreto, la disyunción exclusiva, lo que adicionalmente supone una sorprendentemente sencilla definición alternativa a la suma de las categorías parcialmente aditivas definida por Manes y Arbib.

Con todo esto, se aportan herramientas conceptuales para entender mejor y resolver más eficientemente los problemas que tiene planteados la lógica de la programación, pues se dispone de un punto de vista distinto y nuevo y de toda una familia de instrumentos adicionales.

TRATAMIENTO N-CATEGORIAL DE LA LOGICA DE LA PROGRAMACION

AN N-CATEGORIAL TREATMENT OF THE LOGIC OF PROGRAMMING

In the context of program validation techniques, Hoare's systems for programming and Dijkstra's development of the logic of programming, based on the operator wp, we have endowed the essential features of this development with a structure that permits to study them in the frame of category theory.

In this thesis we show that the set of preconditions of an annotated program segment is an N-category, and the same happens for the set of postconditions. Even more, it is shown that the operator wp acts as a functor between those pairs of N-categories.

Furthermore, guard sets come out in a natural way in the logic of programming, they have been considered from a denotational semantics point of view (Wirth for Pascal and afterwards Scott and al. in a more general setting) and they have been embodied by Manes and Arbib in their partially additive semantics. Then, it is shown in this thesis that the above mentioned guard sets also have the structure of an N-category and that any N-category

with an appropiately defined sum has the structure of a guard set in such a way that, besides, the preorder defined in the guard set by the sum operation coincides with the arrows of the N-category. This sum is just the exclusive or of Boolean Logic and this fact adds a surprisingly simple alternative definition for the sum operation in Manes and Arbib partially additive categories.

The present work, in summary, makes a contribution of conceptual tools for a better understanding and a more efficient solution of the problems posed to the logic of programming and it does so by providing a new different point of view and a whole family of additional techniques.

Lo esencial para un científico es idear una teoría. No hay método para mecanizar ese proceso. Siempre se requiere de la imaginación humana y ciertamente en ciencia se otorga la máxima consideración a la creatividad, a la originalidad. No ensalzamos a los científicos por tener razón. Nadie tiene siempre la razón. Les dignificamos por ser originales, por ser estimulantes, por abrir nuevos caminos de trabajo.

Hermann Bondi

P R E F A C I O

El presente trabajo es el resultado de la reflexión de un informático sobre los métodos técnicos utilizados en la construcción y validación de programas a lo largo de muchos años.

Aunque ya en 1971 Wirth [WIR71] escribió su artículo sobre el desarrollo de programas, y en 1972 Dijkstra sus "Notas acerca de la Programación estructurada" [DIJ72], todas estas ideas y técnicas han ido "calando" en el quehacer cotidiano de los informáticos muy lentamente. Hoy, ya están asimiladas (y sobrepasadas en algunos casos) como métodos cuasi-artesanales para la optimización (o mejora, al menos) de la programación: no existen en nuestra opinión, sin embargo, métodos teóricos desarrollados con rigor científico de utilización práctica universal probada, de programación automática ... ideal de toda persona que dedique tiempo a concebir y desarrollar software.

Sin embargo, en nuestro trabajo profesional, ligado a la docencia en el área de la Metodología de la Programación primero, y de la Lógica Formal después, ha sido una constante

el interés por la definición de marcos conceptuales válidos para la representación de los procesos a implementar y por el establecimiento de técnicas de desarrollo de estos procesos.

En este contexto, nuestra participación en el grupo de Lógica Formal y Teoría de la Computabilidad que trabaja con el profesor Laita y las ideas surgidas de la relación de las N-categorías (y de la teoría de categorías, en general) con las técnicas de validación de programas y con los desarrollos basados en la Lógica de la Programación, han sido sumamente interesantes e ilustrativas: a los primeros estudios e investigaciones, algo tediosos y prolijos (aunque esperanzados en todo momento, por fortuna) ha sucedido el entusiasmo por los resultados que, al menos desde el punto de vista teórico, dan cuenta unificada de cantidad de ideas presentadas desde distintos ángulos por los investigadores de prestigio en éste área, como desarrollaremos en la memoria que sigue.

Esperamos que nuestro afecto y satisfacción (sana y modesta satisfacción del "obrero de la reflexión" por la obra terminada) sean compartidos por quien lea esta memoria y que pueda ser de interés para otras personas y de punto de partida para ulteriores trabajos: como dice Bondi, confiamos en que

estas ideas abran nuevos caminos de trabajo y resulten estimulantes en esa dirección, para alguien.

Debemos por otro lado, agradecer vivamente a los profesores Ledesma y Laita (director y codirector de este trabajo respectivamente) su interés y dedicación, sus ideas, comentarios, críticas ... sin su colaboración, este trabajo, probablemente, no habría visto la luz ó, al menos, no en esta forma. Compañeros y mentores así desearíamos (y hubiéramos deseado) para toda nuestra actividad profesional, pero por desgracia no se encuentran fácilmente.

Queremos agradecer, asímismo, su interés a nuestros compañeros profesores del Departamento de Métodos Estadísticos I. O. e Inteligencia Artificial de la Facultad de Informática en la U. P. de Madrid y a los participantes en el "Seminario del Prof. Trillas" en la Facultad de Informática de Madrid, cuyas enseñanzas y comentarios nos han sido de tanta utilidad.

He estado tentado de no mencionar a mi mujer, Blanca Arbizu, para no caer en el tópico: sería injusto, sin embargo, que no dejara constancia de su apoyo y comprensión en todo el tiempo que he dedicado a esta actividad; sin su impulso constante

probablemente este trabajo no se habría concluido. También deseo agradecer a mis hijas, Blanca, Arantxa y Amaya su comprensión y su aceptación del hecho de que, durante una época, hayan ocupado el tiempo que habitualmente dedico a ellas, las N-categorías.

C O N T E N I D O

0. PRESENTACION Y METODO

El presente trabajo parte del análisis de las técnicas utilizadas en general en Lógica de la Programación y en particular en la metodología que, Dijkstra primero, y Gries posteriormente, han desarrollado para la validación y construcción de programas mediante el operador wp. Este análisis, efectuado desde el punto de vista de la teoría de categorías, permite poner de manifiesto la estructura de los elementos básicos de los sistemas lógicos utilizados y aportar herramientas conceptuales adicionales para su tratamiento. Además, permite dar cuenta unificada de las estructuras (diversas en apariencia) de diferentes sistemas, para una mejor y más fructifera comprensión de ellos.

Tal como subraya Mac Lane [MCL72], "la teoría de categorías arranca de la observación de que numerosas propiedades de los sistemas matemáticos pueden ser unificadas y simplificadas mediante su representación bajo la forma de diagramas de flechas". Naturalmente, este proceso de abstracción e incorporación de ciertas estructuras en otras mas generales que

las comprendan no se produce, en la construcción del conjunto de "la Matemática", por primera vez con la inclusión del concepto de categoría... pero alcanza en él una de sus mas interesantes manifestaciones.

Ya desde los primeros años 60 [MIT65] y [FRE64] en que se formuló la teoría de categorías, ha sido utilizada ampliamente para el análisis de diversas estructuras algebraicas, aunque solo unos años después (1975) Elgot [ELG75] hizo públicas sus observaciones sobre cómo una extensa variedad de diagramas de flujo (sin bucles) se pueden expresar bajo la forma de cualquier categoría con coproductos finitos.

Por aquellas fechas (1974) Laita [LAI74] publicó sus conclusiones sobre la aplicación de la Teoría de Categorías a la Lógica Algebraica, de tal modo que se podían estudiar las fórmulas lógicas (cálculos proposicional y de predicados) como construcciones en dicha Teoría de Categorías.

En el ámbito de la metodología de la programación, se han presentado numerosos planteamientos y desarrollado variadas técnicas para establecer la corrección de los programas, por un lado, y por otro para poder obtener, según una metodología

precisa, las precondiciones correctas a partir de un fragmento de programa dado y de una postcondición concreta.

Una primera aproximación a la resolución de estos problemas se ha producido mediante la "semántica de aserciones" desarrollada a partir de los trabajos de Floyd [FLO67], Hoare [HOA69] y Wirth [WIR71]. Este punto de vista basa la prueba de validez en la demostración de que el programa cumple las especificaciones necesarias para que, partiendo de los datos de entrada (precondiciones) se concluya en los datos de salida (postcondiciones); ó, en otros casos, en demostrar que las precondiciones cumplen las especificaciones deducidas de las postcondiciones dadas y del programa (ó fragmento de programa) concreto que se estudia.

En este mismo entorno de la semántica de aserciones, Dijkstra [DIJ77] ha presentado su sistema basado en el operador wp ("weakest precondition"), que representa para cada programa dado y cada postcondición, la "mínima precondición" que han de cumplir los datos de entrada para que podamos asegurar que el programa concluirá en un estado que satisfaga la postcondición impuesta.

Gries [GRS81] ha desarrollado toda una metodología de la programación basada en la utilización de este operador wp.

Otra aproximación diferente, en principio, viene dada por la semántica denotacional que ofrece técnicas de tipo algebraico para caracterizar la "denotación" de un programa (es decir "la función computada" por el programa), como definen Manes & Arbib [MAN86].

En esta línea, Scott & Strachey [SCO71] presentaron su "semántica del orden" ("order semantics") y Manes & Arbib [libro citado] su "semántica parcialmente aditiva".

La estructura tan interesante del operador wp y su potencialidad en la validación y/ó construcción de programas nos llevó a estudiar más a fondo su relación con las pre y postcondiciones de un fragmento de programa. Se vió que, a partir del operador wp, se pueden estructurar las pre y postcondiciones y que el propio wp actúa como un funtor entre las N-categorías correspondientes, como se desarrolla en esta memoria más adelante.

Además, en la estructuración de la Lógica de la Programación y en la definición en términos precisos (en el contexto del sistema

formal de que se trate) de los esquemas básicos de programación (instrucciones if, do, while, etc) surgen como elementos básicos para la estructuración de los fragmentos de programa, las funciones de guarda. Del mismo modo, surge una situación coincidente al realizar el estudio desde el punto de vista de la semántica denotacional (de "orden", "parcialmente aditiva", etc). La constatación de la importancia de las funciones de guarda y de las funciones guardadas correspondientes, nos indujo a investigar la estructura de estas funciones de guarda y sus interrelaciones en términos algebraicos y/o de teoría de categorías. Como se verá, las funciones de guarda forman una N-categoría y de este hecho se deducen propiedades sumamente interesantes para su aplicación a la Lógica de la Programación.

En la memoria que sigue se presentan ante todo los elementos preexistentes utilizados en nuestro trabajo: teoría de categorías (y, en particular, las N-categorías) y lógica de la programación (operador wp, por un lado y funciones de guarda, por otro, en sus respectivos entornos).

En el segundo capítulo se definen dos sistemas formales de Lógica de la Programación: uno general para el manejo de aserciones de corrección parcial de fragmentos de programa y otro utilizando

el operador wp, válido especialmente para la obtencion de precondiciones a partir de un programa dado y de las postcondiciones impuestas o dadas.

En un tercer capítulo se analiza desde una óptica N-categorial la estructura de los conjuntos de pre y postcondiciones de un programa anotado y se estudia el papel que en la semántica de aserciones juega el operador wp, siempre desde el punto de vista de las N-categorías. Se demuestra que el conjunto de precondiciones (y también de postcondiciones) de un programa puede ser estructurado como una categoría.

En el capítulo cuarto se demuestra, adicionalmente, que el operador wp actúa como un funtor entre categorías, haciendo corresponder los elementos clave de las respectivas estructuras.

En los capítulos quinto y sexto se analiza, también desde una óptica N-categorial la estructura de las funciones de guarda, se demuestra que el conjunto de guardas de un programa tiene estructura de N-categoría y, en sentido inverso, que cualquier N-categoría dotada de una suma adecuada tiene estructura de conjunto de guardas y, además, que el preorden inducido en el conjunto G(X) por la suma coincide con la flecha (morfismos) de

la N-categoría. Por último se ofrece, a través de la disyunción exclusiva, una versión alternativa a la suma de multifunciones que es la clave de los desarrollos presentados por Manes & Arbib en su semántica parcialmente aditiva.

En el septimo se presentan las conclusiones correspondientes, a partir de los resultados obtenidos en los capítulos anteriores, se analizan y discuten, y se proponen las líneas futuras de desarrollo del presente trabajo.

Se concluye la memoria con las referencias bibliográficas de los documentos utilizados en el estudio y preparación de la tesis.

1. ESTADO DE LA CUESTION

Incluimos en este primer capítulo una presentación de los elementos conceptuales que serán utilizados posteriormente en todos nuestros desarrollos.

En una primera sección se presenta la teoría de categorías en sus esquemas y propiedades básicas, así como los elementos característicos a cuyas propiedades dentro de una categoría será necesario aludir después. Naturalmente, no se desarrolla toda la teoría de categorías, pero nos ha parecido útil sintetizar y tener presentes los elementos (y sus propiedades) a los que luego haremos referencia, por comodidad y para que la información concreta utilizable esté autocontenida en esta misma memoria. Se incluyen, así mismo, los últimos desarrollos presentados, de interés para nuestro trabajo.
En una sección posterior se estudian en particular las N-categorías: estructura básica sobre la que se construirán todos los resultados obtenidos.

Por último, incluimos el material necesario de Lógica de la Programación (especialmente en referencia al operador wp, por un

lado, y a las guardas, por otro), presentado desde el punto de vista útil a nuestros propósitos.

1.1. TEORIA DE CATEGORIAS.

Desarrollamos inmediatamente los conceptos de la teoría de categorías que vamos a utilizar después, para definir a continuación (en la sección 1.2) las N-categorías y sus propiedades.

1.1.1. DEFINICIONES.

1.1.1.1. Categoría

Una categoría viene definida por dos grupos de elementos:

a) dos conjuntos: uno, O, de objetos y otro, A, de morfismos definidos entre los objetos; ligados ambos conjuntos mediante dos funciones (dom, cod);
b) dos operaciones: identidad y composición;
de tal modo que:

a) a cada morfismo "f" la función "dom" asigna un objeto o_1 = dom f y la función "cod" asigna un objeto o_2 = cod f; y [F1]

b) la operación "identidad" asigna a cada objeto "o" un morfismo id: $o \to o$ y la operación "composición" asigna a cada par de morfismos (f_1, f_2) con dom f_2 = cod f_1 un morfismo $f_2 \cdot f_1$: dom $f_1 \to$ cod f_2, sujetas estas operaciones a dos axiomas:

Ax1: dados cuatro objetos o_1, o_2, o_3, o_4 y tres morfismos f_1, f_2, f_3, tales que

$o_1 \overset{f_1}{\to} o_2 \overset{f_2}{\to} o_3 \overset{f_3}{\to} o_4$, se cumple que

$$f_3 \cdot (f_2 \cdot f_1) = (f_3 \cdot f_2) \cdot f_1 \qquad [F2]$$

Ax2: dados tres objetos o_1, o_2, o_3 y dos morfismos f_1, f_2, la composición con el morfismo identidad de o_2, ido_2, cumple que

$$ido_2 \cdot f_1 = f_1 \quad y \quad f_2 \cdot ido_2 = f_2 \qquad [F3]$$

El Ax1 asegura que se cumple la propiedad "asociativa" para los tres morfismos dados, mientras la operación de

composición tenga sentido (es decir, mientras los dos morfismos de la igualdad [F2] estén definidos).

El Ax2 indica que el morfismo identidad de un objeto cualquiera cumple el papel de un elemento unidad en la operación de composición, mientras tal composición tenga sentido.

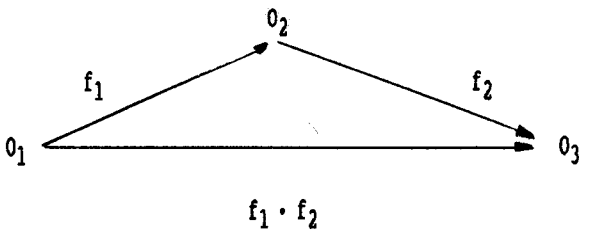
1.1.1.2. Definiciones adicionales

a) Se llama "grafo" (o "diagrama") de la categoría al esquema

$$A \underset{\text{cod}}{\overset{\text{dom}}{\rightleftarrows}} O$$

que representa a los conjuntos y funciones que definen la categoría.

b) Se llama "diagrama de composición" de la categoría al esquema



[F4]

que representa la composición de dos morfismos f_1 y f_2, así como los dominios y codominios involucrados en ella.

c) A los morfismos de la categoría se les llama también "flechas".

Al dominio y codominio de un morfismo se les designa, así mismo, con los nombres de objeto "fuente" u "origen" y "objetivo" o "destino", respectivamente.
Los morfismos se suelen representar, indistintamente, como $f : o_1 \to o_2$ o como $o_1 \overset{f}{\to} o_2$.
La flecha $f_2 \cdot f_1$ se suele llamar "compuesta de f_2 y f_1".

d) En una categoría cualquiera, $A \underset{\text{cod}}{\overset{\text{dom}}{\leftarrow\!\!\rightarrow}} O$, el conjunto de todos los "pares de morfismos componibles" viene definido por
$A \times_O A = \{\langle f_1 , f_2 \rangle \mid f_1 , f_2 \in A \text{ y } \text{dom } f_2 = \text{cod } f_1\}$
y se suele designar como "producto sobre O".

e) Cuando se produce una situación como la descrita en

[F4], es decir, que el resultado de aplicar el morfismo compuesto de $f_2 \cdot f_1$ (de o_1 a o_3) es el mismo que el de aplicar primero f_1 a o_1 y, posteriormente, f_2 a o_2 = cod f_1 = dom f_2, se dice que el "diagrama" correspondiente "conmuta".

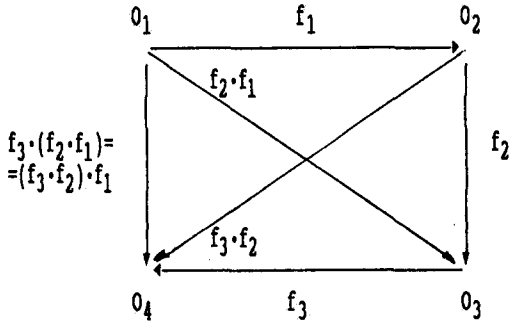
f) En ocasiones, por simplicidad, se obvia la distinción entre los elementos del conjunto O (objetos) y los del conjunto A (morfismos) en una categoría C: se escribe

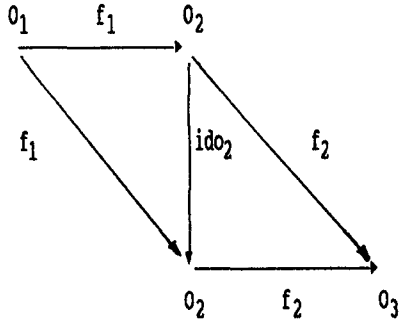
$o \in C$ y también

$f \in C$.

1.1.1.3. Propiedades básicas

a) A partir de la definición e) anterior, los axiomas Ax1 y Ax2 que se cumplen en toda categoría, pueden ser presentados diciendo que conmutan, respectivamente, los dos diagramas





b) Dado que para todo objeto o de una categoría, el morfismo identidad correspondiente ido viene univocamente determinado por las propiedades [F3] del Ax2, a veces se identifica dicho morfismo con el propio objeto y se escribe

$$o = ido = 1o$$

c) A partir de las propiedades [F1] de definición de las operaciones de una categoría, se puede afirmar que

$$dom\ (ido) = o = cod\ (ido)$$

para cualquier objeto o y su correspondiente morfismo identidad ido; y que

$$dom\ (f_2 \cdot f_1) = dom\ f_1 \ ; \quad cod\ (f_2 \cdot f_1) = cod\ f_2,$$

para cualesquiera morfismos componibles f_1 y f_2 en la categoría

$$(o \in O \text{ y } \langle f_2 , f_1 \rangle \in Ax_0A).$$

1.1.1.4. Ejemplos

Como ejemplos de los conceptos anteriores podemos reseñar algunas categorías:

O, categoría vacia (no contiene objetos ni morfismos);

1, categoría formada con un solo objeto, o, y un solo morfismo (la identidad, ido);

2, que contiene dos objetos, o_1 y o_2 y tres morfismos (identidades ido_1, ido_2 y otro tercero $f: o_1 \rightarrow o_2$);

Pfn, categoría de los conjuntos y las funciones parciales que se pueden definir entre ellos (aparece el concepto de dominio de definición DD(f) de cada morfismo $f: o_1 \rightarrow o_2 \mid o_1, o_2 \in O$, tal que $DD(f) \subset o_1$); la operación de composición de morfismos, $f_2 \cdot f_1$, se puede definir, para dos funciones parciales, $f_1 \in Pfn(o_1, o_2)$ y $f_2 \in Pfn(o_2, o_3)$ como $(f_2 \cdot f_1)(x) = f_2(f_1(x))$ para todo $x \in DD(f_2 \cdot f_1)$ y con $DD(f_2 \cdot f_1) = \{x \in o_1 \mid x \in DD(f_1), f_1(x) \in DD(f_2)\}$; el morfismo identidad es una función total;

Mfn, categoría de los conjuntos y las multifunciones entre ellos (una multifunción $f: o_1 \rightarrow o_2$ es una función <u>total</u> de o_1 en el conjunto $P(o_2)$, potencia

de o_2, o conjunto de partes de o_2); la composición $f_2 \cdot f_1$ se define como

$(f_2 \cdot f_1)(x_1) = \{x_3 \in o_3 \mid$ existe un $x_2 \in f_1(x_1)$ que cumple $x_3 \in f_2(x_2)$;

para dos morfismos cualesquiera

$$f_1: o_1 \rightarrow o_2 \quad y \quad f_2: o_2 \rightarrow o_3;$$

se cumple que $Mfn(o_1 , o_2) = TOT(o_1 , P(o_2))$;

TOT, categoría definida sobre los conjuntos, con las funciones totales que se pueden establecer entre ellos (si en las definiciones anteriores sucede que DD(f) = o, para toda f | dom (f) = o | o $\in$ O, las funciones parciales se convierten en totales); los objetos son los conjuntos; los morfismos, las funciones (totales) de unos conjuntos sobre otros; la operación de composición, la usual entre funciones y la identidad, la función que se puede definir de un conjunto sobre si mismo.

1.1.2. FUNTORES

1.1.2.1. Definición

Un funtor es un morfismo entre categorías que define

correspondencias consistentes entre los objetos y morfismos de ambas. Es decir, un funtor F entre las categorías C_1 y C_2 ($F: C_1 \rightarrow C_2$) viene definido por

a) una "función de objetos" (F) que asigna a cada objeto $c_1 \in C_1$, un objeto $c_2 \in C_2$ designado tambien como Fc_1 (es decir, $Fc_1 = c_2$); y **[F5]**

b) una "función de morfismos" (también F) que asigna a cada flecha f_1 de C_1 ($f_1: c_{11} \rightarrow c_{12} \mid c_{1i} \in C_1$) una flecha f_2 de C_2 ($f_2: c_{21} \rightarrow c_{22} \mid c_{2i} \in C_2$ y $c_{2i} = Fc_{1i}$; i=1,2), designada también como Ff_1 (es decir $Ff_1: Fc_{11} \rightarrow Fc_{12}$); de tal modo que:

$F(1_{C1}) = 1_{C2}$ y que **[F6]**
$F(f \cdot f') = Ff \cdot Ff'$ (siempre que la flecha "$f \cdot f'$" compuesta de f y f' este definida en C_1)

Las propiedades indicadas en **[F6]** subrayan la correspondencia que el funtor "F" establece entre el morfismo identidad en C_1 y el correspondiente morfismo identidad de C_2, asi como entre cada par de flechas (f , f') componibles en C_1 y sus correspondiente

morfismos componibles en C_2 (Ff , Ff').

1.1.2.2. Definiciones adicionales y propiedades

a) La definición anterior (párrafo 1.1.2.1.) corresponde al "funtor covariante". Para un funtor de este tipo se subraya su cualidad de covariante cuando se quiere distinguir de los "funtores contravariantes".

Un funtor contravariante responde a la misma definición anterior modificando el apartado b) del siguiente modo:

"b-funtor contravariante) una "función de morfismos" que asigna a cada flecha f_1 de C_1 (f_1: $c_{11} \rightarrow c_{12} \mid c_{1i} \epsilon C_1$) una flecha f_2 de C_2 (designada también como Ff_1) tal que f_2: $c_{21} \rightarrow c_{22} \mid c_{2i} \epsilon C_2$ y $c_{21} = Fc_{12}$, $c_{22} = Fc_{11}$; **[F7]**
o bién

$$Ff_1 : Fc_{12} \rightarrow Fc_{11} ,$$

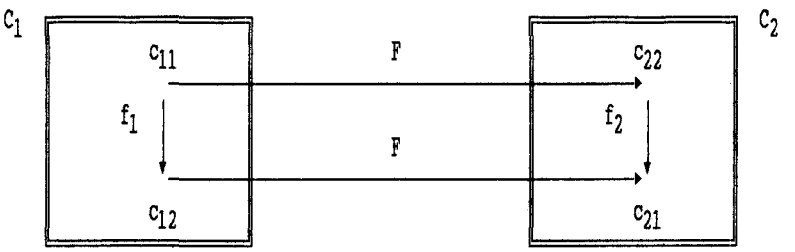
de tal modo que

$$F(1_{C1}) = 0_{C2}$$

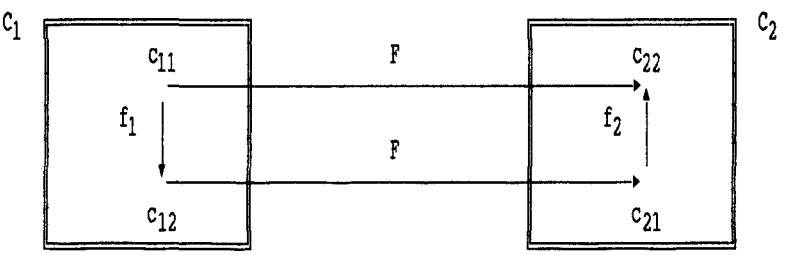
$$F(f \cdot f') = Ff' \cdot Ff$$

(en el mismo supuesto de **[F6]**).

Es decir, si el diagrama correspondiente a un funtor (covariante) es



el diagrama que representa un funtor contravariante es



b) Para un funtor "F" definido entre las categorías C1 y C2 se designa (como en los propios morfismos de las categorías) a C_1 "dominio" del funtor y a C_2 "codominio" del mismo.

c) Dados dos funtores F_1 y F_2, definidos entre tres categorías C_1, C_2 y C_3 de tal modo que

$$C_1 \overset{F_1}{\rightarrow} C_2 \overset{F_2}{\rightarrow} C_3$$

Se llama "funtor compuesto de F2 con F1" (y se representa por F2 · F1) el que asigna a un elemento c ϵ C_1 el elemento $F_2(F1c)$ y al morfismo f ϵ C_1 el correspondiente en C_3: $F_2(F_1f)$.

Esta composición de funtores es asociativa.

Para cada categoría C existe un funtor identidad (Ic: C → C) que cumple el papel de elemento identidad en esta composición.

d) Se llama "isomorfismo" entre dos categorías C_1 y C_2 a un funtor F: $C_1 \rightarrow C_2$ que es una biyección entre ambas (biyección entre los conjuntos de objetos de las categorias y entre los conjuntos de morfismos).

e) Un funtor que "no considera" alguno de los elementos o propiedades de la estructura de los objetos de las

categorias sobre las que se define se suele llamar "funtor olvidadizo" (Ver ejemplo mas adelante, en 1.1.2.3.)

1.1.2.3. Ejemplos

FUNTOR CONJUNTO POTENCIA. Un funtor básico en teoria de conjuntos es el P: CONJUNTO → CONJUNTO, que asigna a cada conjunto A∈CONJUNTO el conjunto P(A), "potencia de A" o "conjunto de partes de A" (formado por todos los subconjuntos de A); y a cada morfismo entre entre conjuntos,

f: A → B | A,B ∈ CONJUNTO, el morfismo Pf: P(A) → P(B) que hace corresponder a un subconjunto $S_A \subset A$ su imagen $S_B = f(S_A) \subset B$.

Este morfismo, P, es en efecto un funtor, puesto que cumple las dos propiedades **[F6]**, ya que

$$P(1_A) = 1_{P(A)} \quad y$$

$$P(f_2 \cdot f_1) = P(f_2) \cdot P(f_1)$$

FUNTOR OLVIDADIZO. El funtor U: GRUPO → CONJUNTO que asigna a cada grupo G el conjunto U(G) = A de sus elementos (obviando la multiplicación y "olvidandose", por tanto, de la estructura de grupo de que está dotado G) y que asigna a cada morfismo f entre grupos (f: $G_1 \rightarrow G_2$) la misma flecha pero considerada como mera función entre conjuntos f: $UG_1 \rightarrow UG_2$, es un funtor olvidadizo.

Así mismo, el morfismo V: ANILLO → ABELIANO, que hace corresponder a cada anillo "A" su grupo abeliano "G", y a cada morfismo f: A → A' entre anillos la misma función considerada como un morfismo aditivo, es un funtor olvidadizo.

FUNTOR CONTRAVARIANTE. En la lógica proposicional, el conjunto de las proposiciones forma una categoría. Sobre esta categoría, la función negación (¬) es un funtor contravariante. En efecto, si sucede que f: A → B, siendo A y B dos proposiciones (objetos de la categoría), se cumple que Ff: FB → FA y no Ff: FA → FB (donde F es el funtor ¬, pues FB = ¬B y FA = ¬A)

1.1.3. TRANSFORMACION NATURAL

1.1.3.1. Definición

Una transformación natural es un morfismo entre dos funtores que permite "trasladar" el resultado de la aplicación de un funtor, al resultado correspondiente con el otro funtor. Es decir, una transformación natural T entre dos funtores F_1 y F_2 ($T: F_1 \rightarrow F_2$), funtores definidos entre dos categorías C_1 y C_2 (F_1 , $F_2: C_1 \rightarrow C_2$), es un conjunto de morfismos en C_2, Tc_i, correspondientes a los elementos $c_i \epsilon C_1$ tales que, dada una flecha en C_1, $f: c_1 \rightarrow c_2$ sucede que

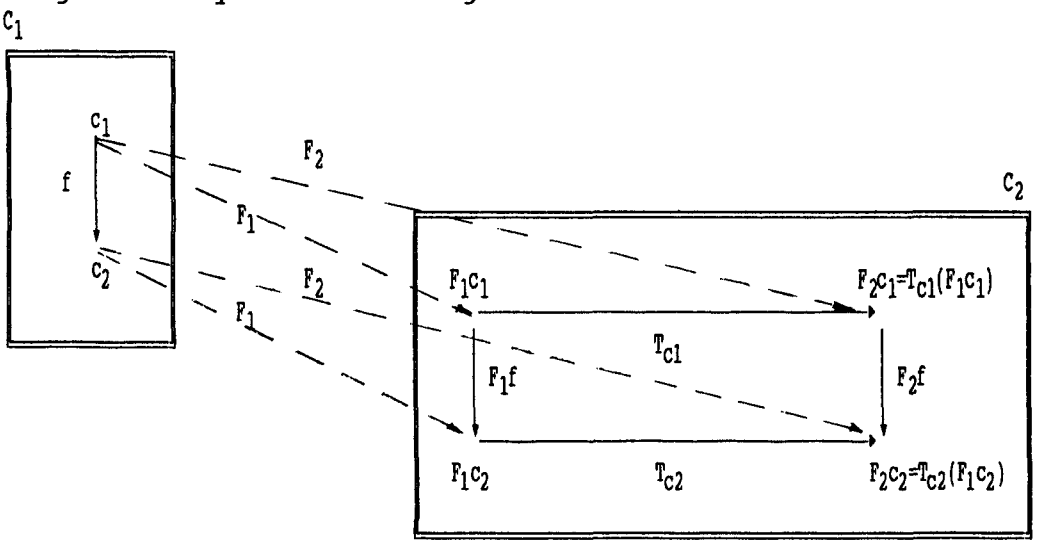
a) $Tc_1(F_1c_1) = F_2c_1$, $\quad Tc_2(F_1c_2) = F_2c_2$ **[F8]**

b) $T(F_1f) = F_2f$, donde

$$F_1f: F_1c_1 \rightarrow F_1c_2 \text{ y}$$
$$F_2f: F_2c_1 \rightarrow F_2c_2,$$

como corresponde a los funtores F_1 y F_2.

Esto significa que en el diagrama



el cuadro de la derecha conmuta.

1.1.3.2. Definiciones adicionales y propiedades

a) Se llama "componentes" de la transformación natural a los morfismos (de C_2), Tc_i, $\forall\ c_i \in C_1$.

b) Se llama "isomorfismo natural" o "equivalencia natural" a una transformación natural tal que cada componente Tc_i es invertible en C_2. Se escribe entonces que T: $F_1 \equiv F_2$ y se dice que los funtores F_1 y F_2 son "naturalmente equivalentes".

c) En general, en una categoría C se dice que un morfismo $f: c_1 \rightarrow c_2$, es invertible (es un isomorfismo) si existe en C otro morfismo f' ($f': c_2 \rightarrow c_1$) tal que

$$f' \cdot f = 1_{c1} \quad y \quad f \cdot f' = 1_{c2}.$$

En este caso se dice que los elementos c_1 y c_2 son "isomorfos" y se escribe $c_1 \equiv c_2$. La relación de isomorfismo es claramente reflexiva, simétrica y transitiva (de equivalencia).

1.1.4. OTROS ELEMENTOS

Además de las estructuras y conceptos anteriores, sobre los que se van a fundamentar nuestros posteriores desarrollos, es interesante reseñar algunos otros elementos y construcciones útiles de la teoría de categorías:

1.1.4.1. Categoría preorden

Una Categoría C es "preorden" si, dados dos objetos cualesquiera de ella c_1 y c_2, existe a lo sumo una flecha entre esos objetos $c_1 \rightarrow c_2$.

Podemos definir una relación de orden "≤" en los objetos de

la categoría diciendo que $c_1 \leq c_2$ si y solo si existe en C un morfismo $c_1 \rightarrow c_2$. La relación es reflexiva (porque para cualquier $c_i \epsilon C$ siempre se puede definir un morfismo $c_i \rightarrow c_i$) y es transitiva (como lo son los morfismos sobre los que está definida la categoría); no tiene por qué ser, necesariamente, ni simétrica ni antisimétrica.

A la inversa, se puede asegurar que cualquier conjunto C sobre el que se haya definido una relación de orden con las propiedades indicadas, es una categoría preorden (Ver detalles en [MCL72], pag. 11).

Es importante subrayar que el concepto de preorden incluye el de "orden parcial" (si se añade la exigencia de que $c_1 \leq c_2$ y simultaneamente $c_2 \leq c_1$, impliquen $c_1 = c_2$) y el de "orden lineal" (un orden parcial en el que dados dos objetos c_1 y c_2, siempre sucede que $c_1 \leq c_2$ o bién $c_2 \leq c_1$); por tanto, la categoría de los conjuntos parcialmente ordenados (en inglés "partially ordered set" o poset) es una categoría preorden; también lo es el conjunto de los números naturales con la relación de orden $\leq$ usual (concretamente $(N, \leq)$ es un conjunto "totalmente ordenado").

1.1.4.2. Objetos inicial y terminal

Se dice que un objeto c_1 de una categoría C es "inicial" si existe exactamente un morfismo de c_1 a cada uno de los restantes objetos de la categoría. Decir que "existe exactamente un morfismo" significa que hay uno y solo uno. Esto se indica con el símbolo de exclamación y escribimos, por tanto,

$$!: c_1 \rightarrow c_2,$$

donde c_2 es un objeto cualquiera de la categoría C.

Es facil comprobar (Ver [MAN86], pag. 48) que el objeto inicial (si existe) de una categoría cualquiera es único (salvo isomorfismos).

Un objeto c_1 de una categoría se llama "terminal" si existe exactamente un morfismo desde cada elemento c_2 de la categoría a él. Escribimos, por tanto,

$$!: c_2 \rightarrow c_1,$$

donde c_2 es un objeto cualquiera de la categoría C.

También el objeto terminal de una categoría, si existe, es único (salvo isomorfismos).

En la categoría de las proposiciones, ya aludida, $A \cdot \neg A$ es el objeto inicial (donde "$\cdot$" representa al producto definido en 1.1.4.3. y "$\neg$" funtor contravariante, tal como hemos indicado en 1.1.2.3.) para cualquier objeto (proposición) de la categoría. Así mismo, $A * \neg A$ es el objeto terminal (donde "$*$" es el coproducto).

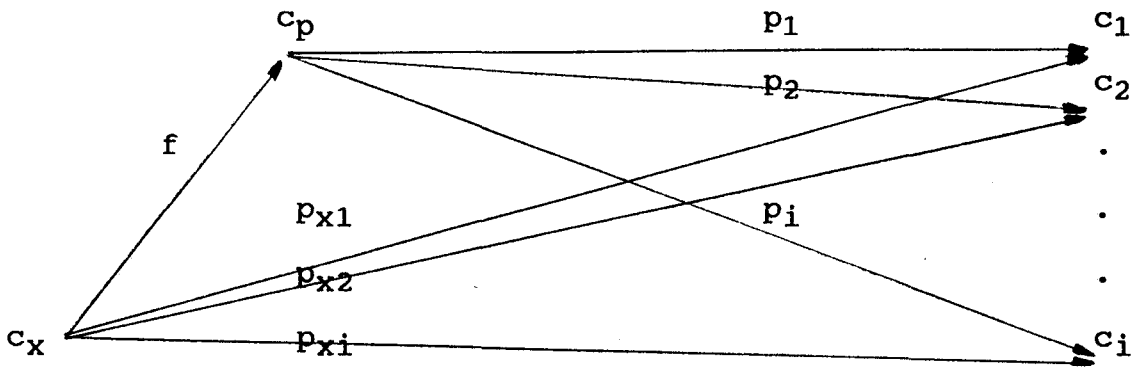
1.1.4.3. Producto y coproducto

Dada una categoría, C, y una familia de objetos, $(c_i \mid i \in I)$, de ella, se define el "producto" de estos objetos como el objeto c_p tal que:

a) existe una familia de morfismos, $(p_i \mid i \in I)$, tales que para cada $i \in I$, $p_i: c_p \rightarrow c_i$; y

b) dado cualquier otro objeto c_x de la categoría para el que exista una familia de morfismos que cumplan una propiedad equivalente a la anterior (es decir, que existan unos $(Px_i \mid i \in I)$ tales que para cada $i \in I$, $Px_i: c_x \rightarrow c_i$), existe un único morfismo $f: c_x \rightarrow c_p$ tal que para todo $i \in I$, $p_i \cdot f = Px_i$ [F9]

Esto se puede representar diciendo que el diagrama



es tal que cada triángulo c_x - c_p - c_i conmuta.

Se puede afirmar que, en una categoría C cualquiera, el producto de una colección cualquiera, (c_i | i ϵ I), de objetos de ella es único, salvo isomorfismo.

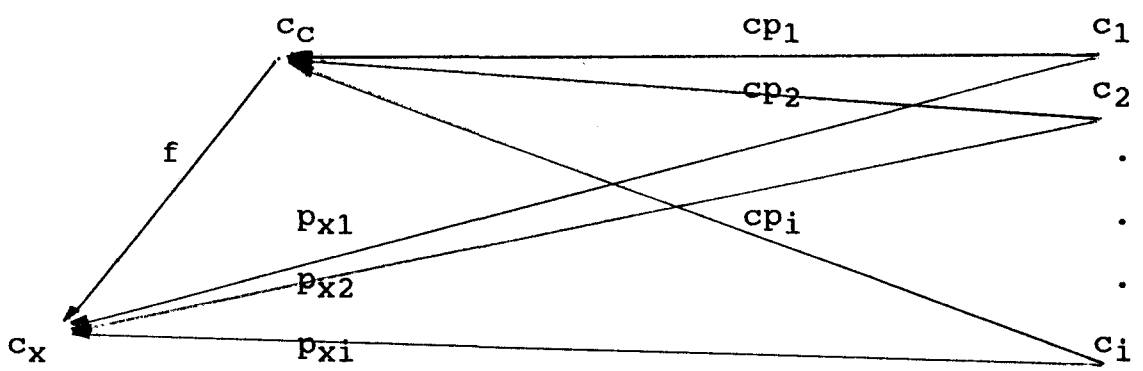
Además, en cualquier categoría, un producto de una familia vacia, es un objeto terminal de la categoría.

COPRODUCTO. Dada una categoría, C, y una familia de objetos, (c_i | i ϵ I), de ella, se define el "coproducto" de estos objetos como el objeto c_C tal que:

a) existe una familia de morfismos, (cp_i | i ϵ I), tales que para cada i ϵ I, cp_i: $c_i \rightarrow c_C$; y

b) dado cualquier otro objeto c_x de la categoría para el que exista una familia de morfismos que cumplan una propiedad equivalente a la anterior (es decir, que existan unos $(p_{xi} \mid i \in I)$ tales que para cada $i \in I$, $p_{xi}: c_i \to c_x$), existe un único morfismo $f: c_c \to c_x$ tal que para todo $i \in I$, $f \cdot cp_i = p_{xi}$. **[F10]**

El diagrama correspondiente (cuyos triángulos conmutan) es:



El coproducto de una familia de objetos es, también, único. El coproducto de una familia vacia, es un objeto inicial de la categoría.

Se dice, por otro lado, que una categoría "tiene productos (coproductos)" si cada familia de objetos tiene un producto (coproducto, respectivamente). También se

dice que la categoría tiene "productos (coproductos) finitos" si cada familia finita (es decir, con I conjunto finito) tiene producto (respectivamente, coproducto).

1.1.4.4. Categorías parcialmente aditivas.

Dentro del contexto de la semántica denotacional, Manes & Arbib [MAN86] definen la estructura de la categoría parcialmente aditiva a partir de un "monoide parcialmente aditivo".

1.1.4.4.1. Un monoide parcialmente aditivo está formado por un par de elementos (C,Σ), donde C es un conjunto no vacío y Σ una función parcial que aplica familias a lo sumo numerables (finitas o infinitas numerables) de elementos de C en elementos de C, de tal modo que se cumple:

a) si $(x_i \mid i \in I)$ es una familia a lo sumo numerable y $(I_j \mid j \in J)$ es una partición de I con J a lo sumo numerable, entonces

$(x_i \mid i \in I)$ es sumable (la suma $\Sigma(x_i \mid i \in I)$ está definida) si y solo si

$(x_i \mid i \in I_j)$ es sumable para cada $j \in J$ y

$(\Sigma(x_i \mid i \in I_j) \mid j \in J)$ es sumable.

En este caso, sucede que

$\Sigma(x_i \mid i \in I) = \Sigma(\Sigma(x_i \mid i \in I_j) \mid j \in J)$;

b) cualquier familia $(x_i \mid i \in I)$ en que I tiene un elemento $(I = \{j\})$ es sumable y $\Sigma(x_i \mid i \in I) = x_j$;

c) si $(x_i \mid i \in I)$ es una familia a lo sumo numerable y la subfamilia $(x_i \mid i \in F)$ es sumable para cada subconjunto F de I, entonces $(\Sigma_i \mid i \in F)$ es sumable.

1.1.4.4.2. Se puede demostrar ([MAN86], pg 74) que si para el monoide parcialmente aditivo (C,Σ) sucede que $\Sigma(x_i \mid i \in I) = 0$ entonces $x_i = 0 \quad \forall i \in I$.

1.1.4.4.3. Una estructura parcialmente aditiva Σ sobre una categoría C es una asignación de una estructura Σ_{XY} a la categoría,de modo que $(C(X,Y),\Sigma_{XY})$ es un monoide parcialmente aditivo para cada par de objetos X , Y de la categoría C, que cumple la propiedad distributiva de la composición respecto de la suma:

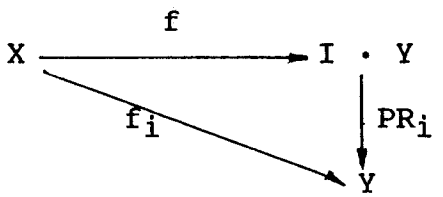
para cualesquiera $f: W \to X$ y $h: Y \to Z$ y para cualquier familia $(g_i \mid i \in I)$ sumable en $C(X,Y)$, sucede que $(g_i f \mid i \in I)$ y $(hg_i \mid i \in I)$ son sumables y, además,

$$(\Sigma_{XY} g_i)f = \Sigma_{WY}(g_i f) \quad \text{y} \quad h(\Sigma_{XY} g_i) = \Sigma_{XZ}(hg_i).$$

1.1.4.4.4. Una categoría parcialmente aditiva es una categoría C con coproductos a lo sumo numerables y una

estructura Σ parcialmente aditiva que satisfacen los dos siguientes axiomas:

a) si $(f_i \mid i \in J)$ es una familia a lo sumo numerable en la categoría C(X,Y) y existe $f: X \to I \cdot Y$ tal que



entonces $(f_i \mid i \in I)$ es sumable.

b) si $f,g: X \to Y$ son sumables, entonces $in_1 f, in_2 g: X \to Y + Y$, son sumables.

1.1.4.4.5. Se puede demostrar **[MAN86]**, que en una categoría parcialmente aditiva, C, dadas $f_i: X \to Y$; $g_i: Y \to Z$, si Σf_i existe, también existe $\Sigma(g_i \cdot f_i)$.

1.2. N - CATEGORIAS

Dentro de la teoría de categorías, son de especial interés a nuestro propósito las N-categorías. En efecto, esta estructura está especialmente indicada para representar relaciones lógicas [LED88] y formará parte del nucleo central de nuestros posteriores desarrollos.

1.2.1. DEFINICION

Una N-categoría es una categoría preorden, C, en la que se haya definido un funtor contravariante $N: C \to C$, tales que (categoría y funtor) cumplan las siguientes propiedades:

a) C tiene un objeto terminal, 1;

b) C tiene coproductos finitos [-,-]; **[F11]**

c) el funtor N^2 es naturalmente equivalente a la identidad de C. Es decir, $N^2c \equiv c$ para cualquier objeto de $c \in C$.

d) existe en C un morfismo $f: c_1 \to c_2$ si y solo si $[Nc_1 , c_2] \equiv 1$, para cualesquiera dos objetos c_1 y c_2 de C.

Se puede observar que el esqueleto de una N-categoría es un Algebra de Boole estableciendo las siguientes equivalencias:

a) el objeto terminal se corresponde con el elemento unidad (elemento neutro de la "segunda" operación del álgebra);

b) los coproductos finitos de objetos de la N-categoría corresponden a los elementos del álgebra resultado de operar dos cualesquiera elementos con la "primera" operación;

c) el funtor contravariante se corresponde con la función de complementación del álgebra: de este modo, la propiedad c) de **[F11]** indica la conocida propiedad de que cualquier elemento de un álgebra de Boole es equivalente al "complementario del complementario" de él mismo;

d) el morfismo descrito en d) de **[F11]** tiene correspondencia con la relación de orden del álgebra.

1.2.2. PROPIEDADES DE LAS N-CATEGORIAS.

Cualquier N-categoría C tiene las siguientes propiedades (ver [LAI87]:

1.2.2.1. Existe un objeto inicial, 0; este objeto es 0 = N1.

1.2.2.2. Tiene también (ver propiedad b de [F11]) productos finitos <a , b>, para cada pareja de elementos a,b ϵ C; se cumple que <a , b> = N[Na , Nb].

1.2.2.3. Se cumple en ella la propiedad distributiva de los productos: es decir, <[a , b] , [a , c]> = [a , <b , c>]

1.2.2.4. Existe en C un pseudocomplemento de a respecto de b, para cada pareja de elementos a , b ϵ C; este pseudocomplemento de a respecto de b es precisamente [Na , b], y es único (salvo isomorfismos) para cada pareja a,b.

1.2.2.5. Admite una extensión completa: es decir, una extensión en la que existen coproductos infinitos.

1.2.2.6. Se cumple en ella que si [a , b] $\equiv$ 1 entonces a $\equiv$ 1 y b $\equiv$ 1.

Además, si C_1, C_1^E y C_2 son N-categorias, C_1^E es una extensión de C_1 y C_2 es completa, se puede afirmar que:

1.2.2.7. Cualquier N-funtor (ver definición a continuación) F, tal que F: $C_1 \rightarrow C_2$, se puede extender a un funtor correspondiente F^E definido como F^E: $C_1{}^E \rightarrow C_2$.
(Un N-funtor F entre dos categorias C_1 y C_2 cuyos funtores respectivos son N1 y N2 - tal que F: $C_1 \rightarrow C_2$ - se llama N-funtor si tiene las tres propiedades siguientes:

a) $F1_1 = 1_2$
b) $F[a , b]_1 = [F_a , F_b]_2$ **[F12]**
c) $FN_1 = N_2F$

para dos elementos cualesquiera a,b ϵ C_1, siendo 1_1 y 1_2 los respectivos objetos terminales de C_1 y C_2; y $[- , -]_1$ y $[- , -]_2$ sus coproductos.

1.2.3. N- CATEGORIAS EN LOGICA.

Las propiedades reseñadas en el apartado anterior representan otras tantas propiedades de las Algebras de Boole,

aunque pueden ser presentadas (y justificadas) ciñéndose exclusivamente al entorno de las N-categorías.

Del mismo modo, y dado que se pueden hacer representaciones algebraicas de la lógica proposicional podemos dar también interpretaciones en el contexto de las N-categorias de las fórmulas y axiomas de dicho cálculo de proposiciones; las "versiones" en términos de N-categorías de las expresiones lógicas nos serán de utilidad en los desarrollos posteriores.

Parece inmediato, en función de las definiciones y propiedades dadas, hacer la transcripción de los símbolos lógicos del siguiente modo:

$\vee$ por [- , -]

$\wedge$ por <- , -> **[F13]**

$\neg$ por N

Surge una pequeña dificultad con la implicación, pues mientras en Lógica el resultado de unir dos fórmulas (objetos, en N-categorías) mediante una conectiva es otra fórmula, en N-categorías eso sucede con el producto y coproducto (<a , b> y [a , b] son objetos para a,b ϵ C) pero no con la implicación, pues a $\rightarrow$ b es una flecha (=morfismo) en la

categoria y no un objeto. Se puede, sin embargo, obviar facilmente esta dificultad teniendo en cuenta que la fórmula proposicional $a \to b$ es equivalente a $\neg a \vee b$ y, por tanto, se puede transcribir en N-categorías como [Na , b] (precisamente el pseudocomplemento de a respecto de b). **[F14]**

Esta es la razón de que se escriba el pseudocomplemento de a respecto de b, también como a=>b.

De acuerdo con todo lo anterior, podemos escribir, p. e., (sabiendo que $a \to a \wedge a$ es un axioma proposicional) que

$$[Na , \langle a , a \rangle] = 1$$

para cualquier objeto a de una N-categoría. O bien, según la propiedad d) de **[F11]**, que existe un morfismo f_1 tal que

$$f_1 : a \to \langle a , a \rangle ;$$

como, por otro lado, también es axioma que $a \wedge a \to a$, se puede asegurar la existencia del morfismo f_2: $\langle a , a \rangle \to a$; es decir, que a y <a , a> son objetos isomorfos en la N-categoría a la que pertenecen.

Del mismo modo, son isomorfos

$\langle a , b \rangle$ y $\langle b , a \rangle$

[a , b] y [b , a]

[Na , b] y [NNb , Na] , etc

y se puede asegurar la existencia de morfismos desde

<[Na , c] , [Nb , c]> a [N[a , b] , c]

Na a [Na , b]

<[Na , b] , [Na , Nb]> a Na , etc

1.3. LOGICA DE LA PROGRAMACION

1.3.1 SEMANTICA DE ASERCIONES

Los problemas de la corrección de los programas por un lado, y los de la programación automática, por otro, se han abordado desde diversos puntos de vista. Uno de estos es la semántica denotacional [SCO71], [MAN86] consistente en asignar funciones semánticas a los programas y razonar posteriormente acerca de las denotaciones (interpretaciones) resultantes. Se puede adoptar otro enfoque basado en demostrar las propiedades de los programas (y su corrección, principalmente) mediante la utilización de un lenguaje formal diseñado "ad hoc" en el que se representan (como expresiones predicativas) el propio programa, las condiciones que han de cumplir los datos de entrada y los de salida y sus relaciones.

Este punto de vista (semántica axiomática ó lógica de la programación) parte de la especificación de los programas mediante fórmulas del tipo

$$\{Q\}\ S\ \{R\} \qquad \textbf{[F15]}$$

que representan al programa ó fragmento de programa en sí (mediante la letra S), y condiciones acerca de los estados de computación antes (Q) y después (R) de la ejecución de S.
Un estado viene dado por un conjunto de valores asignados a las variables del programa. Las expresiones {Q} y {R} se llaman aserciones y la expresión **[F15]** se llama programa con aserciones, ó programa anotado.

Una aserción es una expresión predicativa (cuyas variables son las del programa a que se refiere) que define la(s) condicion(es) que ha(n) de cumplir los estados inicial o final del programa: por ello, a la aserción {Q} se le llama precondición y a la {R} postcondición del programa S.

El significado de la expresión **[F15]** es:
"si {Q} es satisfecha por un estado de entrada dado, la ejecución de S (partiendo del anterior estado) concluye en un estado que satisfará {R}".

La expresión aludida [F15] se llama también aserción de corrección del programa y, puesto que puede ser verdadera ó falsa, puede ser considerada, a su vez, como una expresión predicativa en el contexto de un sistema formal de Lógica de la Programación.

Si partimos de la expresión del programa (completo) anotado, normalmente nos será de utilidad dividirlo en fragmentos menores cara a estudiar su corrección: aparece así una cadena de programas (subprogramas a fragmentos de programa; en ocasiones, simples instrucciones individuales) entre los que se intercalan aserciones, de tal modo que cada aserción es precondición del programa siguiente y postcondición del que le precede

$$.....\{Q_1\}\ S_1\ \{Q_2\}\ S_2\ \{Q_3\}\ S_3\ \{Q_4\}..... \qquad \textbf{[F16]}$$

se puede entonces probar la corrección de cada fragmento, en un sistema lógico adecuado, y deducir de ello la validez de la aserción de corrección completa.

El sistema formal utilizado para ello se suele construir a partir de un sistema de primer orden que incluya axiomas para el tratamiento de los números y añadiendo, además, reglas deductivas que aseveren la corrección de las expresiones

predicativas consecuentes a partir de la corrección de las expresiones antecedentes, cuando se ejecuta entre ellas un programa que realice alguna de las diferentes funciones básicas que aparece en cualquier lenguaje de programación (correspondientes a las estructuras de if, do, while, etc).
Ver al respecto [LED89b] y pruebas de la validez y completud de un sistema de este tipo en [DBK80] y [COO78].
De este modo, la corrección de un fragmento de programa anotado {Q}S{R}, se puede **demostrar** deduciendo la expresión predicativa que representa su corrección como una fórmula válida más del sistema formal que se esté utilizando.
La corrección del programa total se demuestra mediante la aplicación de una regla adecuada de concatenación de programas.

1.3.2 EL TRANSFORMADOR DE EXPRESIONES PREDICATIVAS WP.

1.3.2.1. Significado de wp.

En el contexto antes descrito, Dijkstra introdujo [DIJ77] el operador wp ("weakest precondition") para representar, a partir de un programa S y una postcondición R la precondición "menos severa" que ha de cumplir un estado para asegurar la conclusión de S en R.

Gries ha desarrollado [GRS81] un sistema formal basado en la utilización de este operador wp, que sustituye la determinación de la precondición {Q} de la terna {Q}S{R} por la obtención de la wp(S,R) para una R dada y para cada "tipo de programa" S posible en un lenguaje de programación; tomando aquí la expresión "tipo de programa" como estructura básica de programación (es decir, estructura alternativa - if -, estructura repetitiva - loop -, asignación - := -, etc). Se obtiene con ello una potencia de cálculo y una claridad notables.

Por otro lado, como R es una expresión predicativa, si fijamos un tipo de programa ó instrucción de programa, S, el operador wp(S,R) aparece como una cierta función de la expresión predicativa R, $wp_S(R)$, por lo que a wp se le llama también "transformador de expresiones predicativas".

1.3.2.2 Definición.

Podemos definir wp(S,R) como el predicado que representa a todos los estados tales que si S se ejecuta a partir de uno de ellos, seguro que concluye (en un tiempo finito) en un estado que satisface R.

1.3.2.3. Propiedades.

1.3.2.3.1. Si retornamos a la expresión anotada del programa, {Q}S{R}, podemos asegurar que la precondición Q (cualquier precondición Q que satisfaga la aserción {Q}) es una condición más severa ó restrictiva que la representada por wp(S,R): es decir, si un estado (asignación de valores a las variables) satisface Q seguro que satisface la condición wp(S,R); se puede afirmar, en consecuencia, que el conjunto de estados que satisface Q es un subconjunto del conjunto de los estados que satisfacen wp(S,R).

1.3.2.3.2. Como una condición (pre ó postcondición) en una aserción de corrección de programas, es una proposición (de una N-categoría), la propiedad anterior de relación entre condiciones (1.3.2.3.1) puede ser representada mediante la inclusión, por lo que podemos escribir que

$$Q \rightarrow wp(S,R) \qquad \textbf{[F17]}$$

para cualquier Q, precondición de S{R}.

1.3.2.3.3. Pero, a su vez, wp(S,R) es una (pre)condición que han de cumplir los estados para asegurar la conclusión en R. Por tanto, la propia wp(S,R) es una proposición ó, desde otro punto de vista (1.2.3), un objeto de la N-categoría C.

1.3.3. LAS FUNCIONES DE GUARDA

En el análisis de las estructuras básicas de programación (instrucciones individuales ó pequeños fragmentos de programa que realizan funciones básicas de computación) surgen varias instrucciones elementales a partir de las cuales se construyen las restantes; fundamentalmente: asignación, ejecución guardada, salto ó bifurcación y concatenación. En la construcción de la lógica de la programación con el operador wp se dá cuenta de todas ellas, aunque para el salto (especialmente cuando aparece en estructuras repetitivas ó "loops") las soluciones propuestas (véase especialmente [GR81]) no son del todo satisfactorias.

Nosotros nos hemos centrado en el análisis de las instrucciones de ejecución guardada por su relevancia en el contexto y su tan interesante estructura. En efecto, las

instrucciones de ejecución guardada aparecen en casi todas las principales estructuras básicas de programación:

a) la construcción básica alternativa viene representada en los lenguajes de programación procedurales (MODULA-2, PASCAL, ALGOL...) por la instrucción IF. Una expresión de esta instrucción puede ser "if A then B else C" que corresponde a la estructura castellana

si A, ejecutar B; si no A, ejecutar C

donde A debe ser una condición y B y C fragmentos de programa (ó instrucciones en el caso más simple) a ejecutar;

b) la construcción alternativa múltiple (correspondiente al CASE, SELECT, etc., dependiendo del lenguaje utilizado) responde a la estructura

si A_1 ejecutar B_1

si A_2 ejecutar B_2

si A_3 ejecutar B_3

.

.

.

si A_n ejecutar B_n

c) e, incluso, una instrucción más compleja como el WHILE, responde a la estructura:

(1) si $A_1 \vee A_2 \vee A_3$ entonces

si A_1 ejecutar B_1

si A_2 ejecutar B_2

.
.
.

si A_n ejecutar B_n

IR A (1)

si $\neg A_1 \wedge \neg A_2 \ldots \neg A_n$ entonces FIN.

En todas estas estructuras aparece como elemento fundamental la "instrucción de ejecución guardada", que responde al esquema

si A, ejecutar B

ó, con notación lógica, $A \rightarrow B$, donde A debe ser una condición y B una función a ejecutar.

Se puede considerar la expresión A como una función entre el conjunto de estados (posibles valores asignados a las variables) y el conjunto {V,F}: es decir, cada estado hará verdad ó nó la condición impuesta por A.
En este sentido, la expresión A se llama "guarda" (ó "función de guarda") de la instrucción de ejecución guardada y B se llama instrucción guardada por A. Estas guardas tienen, como se vé, una importancia fundamental en la construcción de estructuras de programación y son muy importantes, en la

Lógica de la Programación, sus interrelaciones y su posible estructuración interna. Por ello precisamente, éste es uno de los aspectos desarrollados con detalle en la presente memoria.

1.3.4. LAS FUNCIONES DE GUARDA EN LA SEMANTICA DENOTACIONAL Y LA SUMA DE MULTIFUNCIONES.

1.3.4.1. Funciones de Inclusión ó Guardas. Desde el punto de vista de la semántica denotacional, Manes & Arbib utilizan [MAN86] las funciones parciales (y las multifunciones) como las construciones básicas para identificar fragmentos de programa dentro de la semántica parcialmente aditiva que desarrollan.

Aunque se pueden establecer desarrollos paralelos (no idénticos) para el caso de las funciones parciales (Pfn (X , Y)) y las multifunciones (Mfn (X,Y)), es posible también definir una relación entre ambos conjuntos haciendo corresponder a cada función parcial $f_p \in$ Pfn (X,Y) una multifunción $f_n \in$ Mfn (X,Y) según la siguiente definición

$$f_n(x) = \begin{cases} \{ f_p(x) \} , & \forall x \in \text{Dominio}(f_p) \\ \Phi , & \text{en caso contrario.} \end{cases} \qquad \textbf{[F18]}$$

De este modo, los desarrollos obtenidos para las multifunciones son válidos en general para las funciones parciales, excepto en algunos aspectos críticos (basicamente en cuanto a los dominios de definición de unas y otras).

Para los fines que nos interesan, la conversión anterior de funciones parciales en multifunciones es interesante, pues nos remite a un entorno mas amplio de funciones que representa fragmentos de programa con estructura mas general.

Nos ceñiremos pues a considerar en nuestro estudio las multifunciones (incluyendo en ellas a las funciones parciales), subrayando, cuando sea necesario, las salvedades pertinentes.

Dentro de las multifunciones, son de gran interés las funciones de inclusión, en especial cuando aparecen compuestas con otras multifunciones: en ese caso se denominan "funciones de guarda" o, simplemente, "guardas".

En efecto, en la categoría de las multifunciones, podemos considerar un conjunto cualquiera X y en él un subconjunto

A. La función inclusión de A es el morfismo $inc_A \in Mfn(X, X)$ definido como

$$inc_A(x) = \begin{cases} \{x\}, \text{ si } x \in A \\ \\ \Phi, \text{ en caso contrario} \end{cases} \qquad [F19]$$

(observese que se ha incluido la segunda parte de la definición de $inc_A(x)$ para completar la definición y poder considerar $inc_A \in Mfn(X, X)$ según [F18]).

Cuando aparece la función compuesta f•g, en la que f es una multifunción cualquiera y g es una función de inclusión, se dice que g es una función de guarda (o guarda, simplemente) de f, porque efectivamente, " si g es verdad se ejecuta f, en caso contrario el resultado es indefinido": es decir, g "guarda" el paso a la ejecución de f. Se suele escribir la composición f•g como $g \rightarrow f$ para subrayar esta cualidad de guarda de la función g.

1.3.4.2. Suma de multifunciones.

Dados los conjuntos X e Y y un conjunto de índices I. Para cada $i \in I$ se considera la función $f_i \in Mf_n(X, Y)$; se dice

que $(f_i \mid i \in I)$ es una familia indexada en I de Mfn(X , Y).

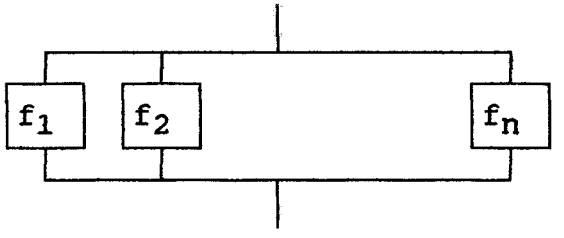
A partir de una familia de multifunciones como la indicada, podemos definir la suma

$$\Sigma(f_i \mid i \in I)(x) = \bigcup_{\forall i \in I} f_i(x) = \{y \in Y \mid y \in f_i(x) \text{ para algún } i \in I\} \qquad \text{[F20]}$$

esta suma se escribe también $(\Sigma_{i \in I} f_i)(x)$, o si el conjunto I es finito

$$(f_1 + f_2 + \ldots + f_n)$$

o, incluso, Σf_i si el conjunto I al que pertenecen los indices i se deduce del contexto. Cuando el conjunto I es finito, la suma se puede representar por el diagrama



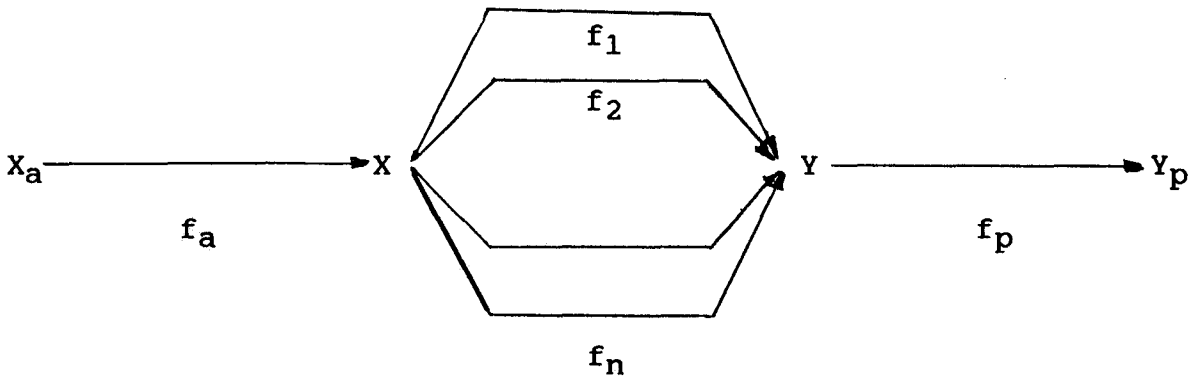
Se puede demostrar que la ley de composición de las multifunciones es distributiva respecto de la suma que acaba de ser definida.

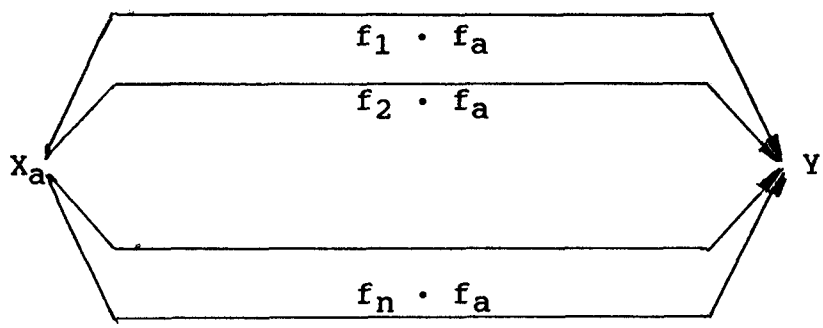
En efecto, si tenemos una familia de multifunciones

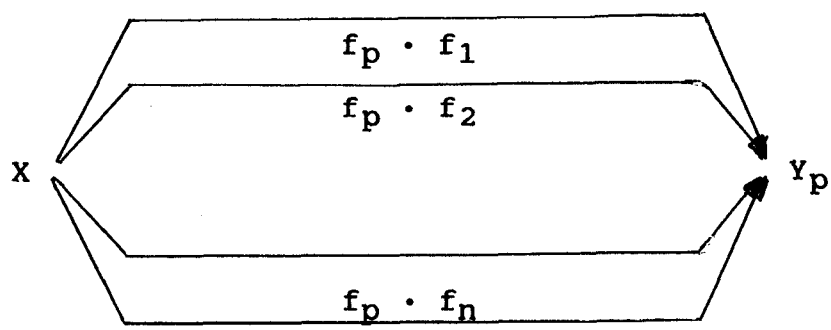
$(f_i \mid i \in I)$ tal que $f_i \in Mfn(X , Y)$ y dos morfismos mas $f_a \in Mfn(X_a , X)$ y $f_p \in Mfn(Y , Y_p)$ se cumple que

$$(\Sigma f_i) \cdot f_a = \Sigma(f_i \cdot f_a) \in Mfn(X_a , Y) \qquad \textbf{[F21]}$$

$$f_p \cdot (\Sigma f_i) = \Sigma(f_p \cdot f_i) \in Mfn(X , Y_p)$$







puesto que

si $y \in ((\Sigma f_i) \cdot f_a(x_a))$ para algún $x_a \in X_a$ y algún $y \in Y$ entonces $\exists x \in f_a(x_a) \mid y \in (\Sigma f_i)(x)$ según **[F1]** y, por tanto $\exists x \mid x \in X$ y para algún $i \in I$, $y \in f_i(x)$ por **[F20]**; es decir $\exists i \in I \mid y \in (f_i \cdot f_a)(x_a)$ y, en consecuencia,

$$y \in (\Sigma(f_i \cdot f_a)(x_a)) \quad \text{c.q.d.;}$$

por otro lado,

si $y_p \in (f_p \cdot (\Sigma f_i))(x)$ para algún $x \in X$ y algún $y_p \in Y_p$ entonces $\exists y \in (\Sigma f_i)(x) \mid y_p \in f_p(y)$, según la definición **[F1]** y en este caso $\exists i \in I \mid y \in f_i(x)$, según **[F20]** y, también, $\exists i \in I \mid y_p \in (f_p \cdot f_i)x$ y se puede concluir que $y_p \in \Sigma(f_p.f_i)(x)$ c.q.d.

1.3.4.3. Propiedades de las funciones de inclusion.

Se puede afirmar (ver [MAN86]) que

1.3.4.3.1. inc_Φ es una función especial, nula, tal que

$$inc_\Phi (x) = \Phi, \text{ pues no existe } x \in \Phi$$

1.3.4.3.2. $inc_X = id_X$ es la función identidad de X.

También se puede afirmar que para dos subconjuntos cualesquiera A, B ϵ X se cumple que

1.3.4.3.3. $(inc_A + inc_B)\ (x) = inc_A\ (x)\ U\ inc_B\ (x) =$

$$= \begin{cases} \{x\}, \text{ si } x \text{ pertenece a A, B o ambos} \\ \\ \Phi, \text{ si } x \notin A\ U\ B \end{cases}$$

(donde + es la suma parcial definida por Manes & Arbib [MAN86])

1.3.4.3.4. $inc_A \cdot inc_B = inc_{A \wedge B} = inc_B \cdot inc_A$

(donde $\cdot$ es la composición de funciones usual en cualquier categoria, y $\wedge$ la intersección de conjuntos)

1.3.4.3.5. El conjunto A $\subset$ X define en X un conjunto complementario A' $\subset$ X.

De la definición de las funciones de inclusión y de las propiedades de los conjuntos se deduce que

$inc_A\ (x) + inc_{A'}\ (x) = 1\ (= id_X)$

$inc_A\ (x) \cdot inc_{A'} = inc_{A'} \cdot inc_A = 0\ (= inc_\Phi)$

1.3.4.3.6. $Inc_{(A')'} = inc_A$

2. DOS SISTEMAS FORMALES PARA LA LOGICA DE LA PROGRAMACION.

Tal como hemos indicado (ver 1.3.1] se puede comprobar la corrección de un programa (ó fragmento de programa) demostrando la validez de la aserción que representa al programa (con sus pre y postcondiciones) como una fórmula predicativa dentro de un sistema formal definido al efecto.
Queremos en este capítulo estudiar un sistema formal de este tipo, y posteriormente, dar la versión del mismo utilizando el operador wp.

2.1. SISTEMA FORMAL DE PRIMER ORDEN PARA LA LOGICA DE LA PROGRAMACION.

El sistema que proponemos, parte de la definición previa supuesta del significado semántico de un conjunto de instrucciones de programación correspondientes a un lenguaje que disponga de una sintaxis semejante a la del MODULA-2 ó al PASCAL. (En el siguiente apartado presentamos dicho lenguaje). Las estructuras básicas de programación en un lenguaje de este tipo pueden ser construidas casi exclusivamente a partir de cuatro funciones

elementales: asignación, ejecución guardada, salto ó bifurcación y concatenación. A partir de estas funciones elementales se pueden formar estructuras más complejas como construcciones alternativas ó iterativas, bucles, etc. La sintaxis concreta del lenguaje utilizado no es de interés en este momento y se supone conocida.

2.1.1. LENGUAJE UTILIZADO.

Se utilizarán las siguientes instrucciones de programación:

2.1.1.1. Asignación.

Se atribuye a la variable x el valor de la exprexión e:

x := e

Suponemos que la expresión e está definida antes de la ejecución del comando de asignación.

2.1.1.2. Ejecución guardada.

Se ejecutará la instrucción B si se cumple la condición A.

Si A, ENTONCES B.

También supondremos que la condición A está definida (vale V-Verdadero- ó bien F-falso-). La instrucción B es una instrucción cualquiera: una asignación, otra ejecución guardada u otra expresión compleja cualquiera.

2.1.1.3. Salto.

No ejecutar ninguna instrucción y pasar a ejecutar, a continuación, la instrucción que indica la etiqueta

IR A Etiqueta

donde "Etiqueta" es el nombre asignado a una instrucción.

2.1.1.4. Concatenación.

Ejecución de un comando que equivale a la ejecución sucesiva de dos instrucciones, de tal modo que el estado de salida de la primera sea de entrada para la segunda. Si las dos instrucciones de que se trata son S_1 y S_2, se representa la concatenación de ambas como

$$S_1 \; ; \; S_2$$

A partir de estas definiciones se pueden establecer estructuras más complejas de programación: se pueden definir estructuras alternativas (instrucciones IF, CASE, etc), instrucciones de iteración (correspondientes a LOOP, FOR, WHILE, etc), etc.

a) Por ejemplo, se puede analizar la estructura de la instrucción IF (de MODULA-2) siguiente:

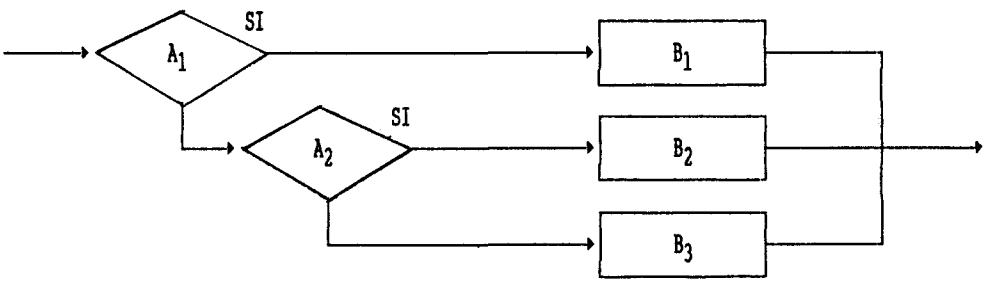
IF A_1 THEN B

ELSIF A_2 THEN B_2

ELSE B_3

END

que se corresponde con el diagrama:



y con la secuencia de instrucciones de ejecución guardada

SI A_1, ENTONCES B_1

SI $\neg A_1 \wedge A_2$, ENTONCES B_2

SI $\neg A_1 \wedge \neg A_2$, ENTONCES B_3

b) Otro caso puede ser la instrucción LOOP con comando EXIT, siguiente:

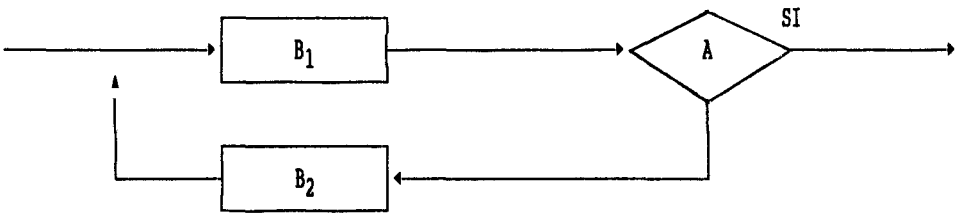
LOOP B_1;

IF A THEN EXIT END;

B_2

END

cuyo diagrama es



equivalente a

Bucle: B_1;

SI A, ENTONCES FIN

SI $\neg$ A, ENTONCES B_2;

IR A Bucle

2.1.2. SISTEMA FORMAL

El sistema formal se construye a partir de un sistema aritmético de primer orden (un sistema de lógica de primer orden que incluya como axiomas no lógicos los de manejo de la aritmética de los números). A tal sistema se le incorporan como reglas de deducción las siguientes:

2.1.2.1. Regla de Asignación.

$$\frac{A_1 \rightarrow (A_2)_x[t]}{\{A_1\}\ x := t\ \{A_2\}}$$

puesto que $\{(A_2)_x[t]\}\ x := t\ \{A_2\}$ es siempre correcta, donde $A_x[t]$ es el resultado de sustituir x por t en A.

2.1.2.2. Ejecución Guardada.

$$\frac{\{A \wedge A_1\}\ B\ \{A_2\}}{\{A_1\}\ \text{IF A THEN B}\ \{A_2\}}$$

Obsérvese que, en esta expresión simple, no se afirma nada del caso en que A no sea cierta (parte ELSE usual de una instrucción de programación). Aquí A es la función de guarda de la instrucción B.

2.1.2.3. Salto.

Para la instrucción de salto no se provee ninguna regla, pues en el sistema formal se manejan aserciones respecto de los estados de las variables y en un salto ó bifurcación no se modifican, por definición, dichos estados. Evidentemente, se puede escribir la aserción (tautológica)

{A} IR A Etiqueta {A}

Cuando se produce un salto hacia atrás (de tal modo que se repita ó pueda repetir la ejecución de una instrucción del programa) ó en general, cuando se ejecutan instrucciones dentro de un bucle, hay que controlar unicamente que la ejecución de las instrucciones de ese bucle concluya. Para ello se introduce alguna variable de control, como veremos a continuación al estudiar estructuras de programación más complejas.

Hay tres "saltos" singulares a los que conviene aludir:

a) SKIP. Esta instrucción no realiza acción alguna, sino que transfiere el control para que se ejecute la instrucción siguiente en secuencia.

b) EXIT. Utilizada en MODULA-2 para "salir" de un bucle LOOP: produce la detención de la ejecución de las instrucciones del bucle y el que se siga con la ejecución en secuencia del resto del programa.

c) HALT. Instrucción para detener la ejecución del programa: equivale, desde el punto de vista del sistema formal, a un salto hacia una instrucción de detención del proceso; pero tampoco se produce ningún cambio en el estado de las variables.

2.1.2.4. Concatenación.

$$\frac{\{A_1\}\ S_1\ \{A_2\},\ \{A_2\}\ S_2\ \{A_3\}}{\{A_1\}\ S_1;S_2\ \{A_3\}}$$

donde A_2 representa el estado intermedio, postcondición de S_1 y precondición de S_2. Esta expresión se puede

generalizar de un modo simple, a la ejecución sucesiva (concatenación) de varios subprogramas.

También se incluyen tres reglas operacionales que facilitan los procesos deductivos:

2.1.2.5. Implicación en postcondición.

$$\frac{\{A_1\}\ S\ \{A_2\},\ A_2 \rightarrow A_3}{\{A_1\}\ S\ \{A_3\}}$$

Como se vé, es una regla meramente instrumental que no representa ninguna estructura concreta de programación.

2.1.2.6. Implicación en Precondición.

$$\frac{A_1 \rightarrow A_2\ ,\ \{A_2\}\ S\ \{A_3\}}{\{A_1\}\ S\ \{A_2\}}$$

2.1.2.7. Cambio de Variable.

$$\frac{\{A_1\}\ S\ \{A_2\}}{\{(A_1)_x[y]\}\ (S)_x[y]\ \{(A_2)_x[y]\}}$$

con tal que y no esté libre en A_2, ni suceda en S.

Para la deducción de aserciones de corrección parcial de

programas ó fragmentos de programas, se pueden utilizar las reglas anteriores (a partir de axiomas y teoremas del sistema aritmético de primer orden en el que se trabaja) ó bien construir reglas más complicadas para las estructuras básicas de programación del lenguaje que se esté utilizando, como hacemos a continuación para dos casos concretos de MODULA-2, a modo de ejemplo:

2.1.2.8. Si se examina una estructura más compleja como la presentada en el apartado 2.1.1., caso a) se puede afirmar que:

$$\frac{\{A_0 \wedge A_1\}\ B_1\ \{C\}\ ,\ \{A_0 \wedge (\neg A_1 \wedge A_2)\}\ B_2\ \{C\}\ ,\ \{A_0 \wedge (\neg A_1 \wedge \neg A_2)\}\ B_3\ \{C\}}{\{A_0\}\ \text{IF ELSIF}\ \{C\}}$$

representando por IF ELSIF la estructura

IF A_1 THEN B_1

ELSIF A_2 THEN B_2

ELSE B_3

END.

Obsérvese que $\{(A_1) \vee (\neg A_1 \wedge A_2) \vee (\neg A_1 \wedge \neg A_2)\}$ es tautología, por lo que no hay que tener ninguna consideración especial con el valor (V o F) que adopten A_1

y A_2. En otras estructuras alternativas en que aparecen varias funciones A_i ($i \in I$) "guardando" la ejecución de otras tantas instrucciones B_i, hay que incluir la premisa $\{\vee A_i, \forall\ i \in I\}$, para asegurar que se cumple al menos una de las A_i.

2.1.2.9. Para concluir la descripción del sistema formal examinaremos el caso b) presentado en el apartado 2.1.1.:

LOOP B_1;
IF A THEN EXIT END;
B_2
END

La regla correspondiente a esta estructura es

$$\frac{\{A_0\}\ B_1\ \{A_1\}\ ,\ \{A_1 \wedge \neg A\}\ B_2\ \{A_2\}\ ,\ A_2 \rightarrow A_0,\quad A_1 \wedge \neg A \rightarrow t > 0\ ,\ \{t > 0\}\ x := t;B_2;B_1\ \{t < x\}}{\{A_0\}\ \text{LOOP}\ \{A_1 \wedge A\}}$$

donde LOOP representa el conjunto de la estructura que estamos estudiando.

Además de los estados que intervienen en el cómputo (A_0, estado inicial; A_1, A_2 estados intermedios) y de la condición A, de conclusión del bucle, aparece en la regla la variable "de contorno" ó de control, t, que a través de la

exigencia impuesta de decrecimiento en cada ciclo del bucle $(B_2;B_1)$ nos asegura la conclusión, en tiempo finito, del proceso.

2.2. SISTEMA FORMAL DEL OPERADOR WP.

Como ya hemos comentado ampliamente, la utilización de la "weakest precondition" introduce en la Lógica de la Programación una "herramienta" sumamente interesante para la "programación automática": obtención de precondiciones a partir de un módulo de programa y una postcondición dada. En efecto, se puede obtener la expresión genérica de la wp(B,A) para cada tipo de programa, B, y deducir la wp de todo un programa complejo mediante la aplicación sucesiva de las reglas pertinentes. Además, para la demostración de la corrección parcial de un programa anotado

$$\{A_1\}\ B\ \{A_2\}$$

se puede proceder calculando $wp(B,A_2)$ y, posteriormente, demostrando en el sistema formal que $A_1 \rightarrow wp(B,A_2)$ con lo que se comprueba la validez de la aserción anterior. Es de notar que $wp(B,A_2)$ es una expresión predicativa más (una fórmula) del sistema y puede ser tratada, por tanto, como tal.

Por ello, para completar nuestro estudio de la Lógica de la Programación, presentamos a continuación la versión con wp del sistema formal definido en la sección anterior.

2.2.1. EXPRESIONES BASICAS.

2.2.1.1. Asignación.

Para la instrucción de asignación (x := t) y para una fórmula cualquiera aritmética de primer orden, A_2, la "weakest precondition" es otra fórmula tal que

$$\models wp(x := t, A_2) \longleftrightarrow (A_2)_x[t]$$

pues, en efecto, llegaremos a A_2 después de hacer la asignación, si y solo si partimos de un estado en el que se cumplan, al menos, las condiciones que supone A_2, habiendo sustituido en ella x por t.

Hay que pensar que afirmar la validez de $\{A_1\}\ x := t\ \{A_2\}$ es lo mismo que afirmar la de $A_1 \rightarrow wp(x := t, A_2)$.
Naturalmente, estamos suponiendo que la expresión t está definida en el momento de ejecutar la instrucción de asignación, lo que se corresponde con la realidad, en la mayoría de los casos prácticos.

2.2.1.2 Ejecución guardada.

Del mismo modo, se razona que

$$\models \text{wp("SI A, ENTONCES B"},A_2) \longleftrightarrow A \rightarrow \text{wp}(B,A_2).$$

En la expresión anterior de la instrucción de ejecución guardada simple, no se tiene en cuenta qué sucede si, en el momento de ejecutar la instrucción, no se cumple A. En los casos de construcciones alternativas múltiples hay que imponer restricciones a los valores de las funciones de guarda A_i presentes, como ya hemos comentado (ver 2.1.2.2.).

2.2.1.3. Salto.

Como en la instrucción de bifurcación no se modifica el estado de las variables, la expresión de la wp correspondiente es sumamente simple.

$$\models \text{wp("IR A Etiqueta"},A_2) \longleftrightarrow A_2$$

Como ya se ha indicado, habrá que tener en cuenta, en las construcciones más complejas, las restricciones y controles a imponer para asegurar la terminación de los bucles en un número finito de pasos.

2.2.1.4. Concatenación.

Por las relaciones de implicación existentes entre las precondiciones de un módulo de programa y la wp del mismo, se vé que

$$\models \; wp("B_1;B_2", A_2) \longleftrightarrow wp(B_1, wp(B_2, A_2))$$

2.2.1.5. Relaciones de Distributividad.

Se puede afirmar que son tautologías las expresiones siguientes, deducidas de las propiedades algebraicas de las fórmulas del sistema de primer orden que se utiliza,

$$\models \; wp(B,A_2) \wedge wp(B,A_3) \longleftrightarrow wp(B,A_2 \wedge A_3)$$

$$\models \; wp(B,A_2) \vee wp(B,A_3) \rightarrow wp(B,A_2 \vee A_3)$$

Obsérvese la implicación en un solo sentido de ésta última expresión. Si los procesos son determinísticos también se cumple la implicación en el otro sentido.

3. LA N-CATEGORIA DE LAS PRECONDICIONES DE UNA POSTCONDICION.

Se presenta y discute en este capítulo una interpretación, en el ámbito de las N-categorías, de las propiedades del operador wp ya descrito (véase 1.3.2).

Veamos en primer lugar cómo al conjunto de todas las precondiciones posibles para una postcondición dada R y un programa establecido S, se le puede dotar de estructura de N-categoría, (subcategoría, por otro lado de la N-categoría de las proposiciones). En el capítulo siguiente estableceremos, a partir de estos resultados, algunas propiedades interesantes del operador wp.

3.1. ESTRUCTURACION DEL CONJUNTO DE LAS PRECONDICIONES.

Las precondiciones posibles en una especificación de corrección de un programa dado S, que ha de concluir en un estado que satisfaga la postcondición R, deben representar estados a partir de los cuales, en efecto, se concluya en otros estados que

satisfagan R. En la expresión {Q} S {R}, {Q} viene especificado como una proposición que ha de validar cualquier estado para que pueda ser considerado precondición de S (para llegar a R), según hemos explicado en 1.3.2.
El conjunto de las proposiciones en general presenta estructura de N-categoría (véase 1.2.3).

Si consideramos el operador wp(S,R), puesto que representa a los estados (todos los estados, exactamente) a partir de los cuales se garantiza la conclusión en R, es una proposición a cumplir por dichos estados.

Tenemos, por tanto, en la N-categoría de las proposiciones (a la que daremos en adelante el nombre genérico C) la proposición wp(S,R) como un objeto más.

3.1.1. DEFINICION DE Ewsr.

A partir de este objeto wp(S,R) ϵ C, se puede construir un conjunto (que llamaremos $E_{WSR)}$ del siguiente modo:

E_{wsr} = { c | c → wp(S,R) es una flecha en C } [F22]

La definición del operador wp(S,R) nos indica que cualquier otra precondición posible de S {R} supone una condición "incluida" en wp(S,R); es decir, en la N-categoría C sucede

que, para cualquier q (precondición de S {R} se cumple $p \rightarrow wp(S,R)$; pero ésta es precisamente la condición que cumple el objeto wp(S,R) en el conjunto E_{WSR}: por tanto, el conjunto E_{WSR} es el conjunto de todas las posibles precondiciones de S {R}.

3.2. EL CONJUNTO DE LAS PRECONDICIONES TIENE ESTRUCTURA DE N-CATEGORIA.

3.2.1. EL FUNTOR CONTRAVARIANTE N_{WSR}.

Si definimos el morfismo $N_{WSR} : E_{WSR} \rightarrow E_{WSR}$ tal que

$$N_{WSR}(c) = < Nc , wp(S,R) >$$

donde N es el funtor de la categoría C y <- , -> su producto, $N_{WSR}(c)$ resulta ser un funtor contravariante.

En efecto, si tomamos un objeto cualquiera $c \in E_{WSR}$, su objeto imagen por N_{WSR} cumple $N_{WSR}(c) \rightarrow wp(S,R)$, como corresponde al producto (ver 1.1.4.3.) y pertenece, por tanto, al conjunto E_{WSR}. [F23]

Además, si consideramos dos elementos c_1, $c_2 \in E_{WSR}$ tales que existe un morfismo $m: c_1 \to c_2$ sucederá (ver **[F32]**) que $< c_2 , c_1 > = c_1$ y, por tanto, $Nc_1 = [Nc_1 , Nc_2]$; en consecuencia,

$$< Nc_1 , Nc_2 > = < [Nc_1 , Nc_2], Nc_2 > =$$

$$= [< Nc_1 , Nc_2 > , < Nc_2 , Nc_2 >] =$$

$$= [< Nc_1 , Nc_2> , Nc_2] = \text{(según 1.2.3)} = Nc_2.$$

Así pues,

$<< Nc_1 , wp(S,R) > , < Nc_2 , wp(S,R) >> = < Nc_2 , wp(S,R)>$

lo que indica que existe una flecha (n) en C (y, por tanto, en E_{WSR}) tal que

$$n : < Nc_2 , wp(S,R) > \to < Nc_1 , wp(S,R)>$$

es decir $n : N_{WSR}(c_2) \to N_{WSR}(c_1)$ **[F24]**

Por otro lado, cumple también las propiedades exigidas en **[F7]**:

$N_{WSR}(wp(S,R)) = < Nwp(S,R) , wp(S,R)> = 0 \in E_{WSR}$ y también $N_{WSR}(m_1 \cdot m_2) = N_{WSR}\, m_2 \cdot N_{WSR}\, m_1$, para $m_1: c_1 \to c_2$, y $m_2: c_2 \to c_3$, $c_i \in E_{WSR}$, puesto que $N_{WSR}\, m_2: N_{WSR}\, c_3 \to \to N_{WSR}\, c_2$ y $N_{WSR}\, m_1 : N_{WSR}\, c_2 \to N_{WSR}\, c_1$ cuya composición dá como resultado (por las propiedades de C)

$n : N_{WSR}\, c_3 \to N_{WSR}\, c_1$, lo que equivale a $N_{WSR}\, (m_1 \cdot m_2)$.

Por tanto, de [F23], [F24] y la comprobación de las propiedades anteriores, se deduce que N_{WSR} es un funtor contravariante.

3.2.2. LA N-CATEGORIA E_{WSR}.

A su vez, el conjunto E_{WSR} con el funtor contravariante N_{WSR} tiene, como veremos, estructura de N-categoría, con wp(S,R) como objeto terminal y con los mismos coproductos que cuando sus objetos los consideramos en la N-categoría C de las proposiciones.

En el conjunto E_{WSR}, subconjunto de C (que es N-categoría) se cumplen todas las propiedades estructurales de la N-categoría excepto, a lo sumo, las de pertenencia al conjunto de los objetos significativos; veamos que éstas también se cumplen:

a) Es claro que el objeto terminal es wp(S,R), por la definición de E_{WSR}.

b) Si consideramos dos objetos c_1 , $c_2 \in E_{WSR}$ sucede que

$$c_1 \to wp(S,R) \quad y \quad c_2 \to wp(S,R)$$

lo que significa que

$$<c_1 , wp(S,R)> = c_1 \quad y \quad <c_2 , wp(S,R)> = c_2 ;$$

pero al considerar el producto $<[c_1 , c_2] , wp(S,R)> =$

$= [<c_1 , wp(S,R)> , <c_2 , wp(S,R)>] = [c_1 , c_2]$, por las equivalencias anteriores, y ésta última expresión significa que $[c_1 , c_2] \to wp(S,R)$;

es decir $[c_1 , c_2] \in E_{WSR}$.

c) Si calculamos $(N_{WSR})^2(c)$ para cualquier $c \in E_{WSR}$, se tiene $(N_{WSR})^2(c) = N_{WSR} (<Nc , wp(S,R)>) =$

$= <N<Nc , wp(S,R)> , wp(S,R)> =$ (según 1.2.2.) $=$

$= <[N^2c , N(wp(S,R))] , wp(S,R)> =$

$= <[c , N(wp(S,R))] , wp(S,R)>$ pues en C, $N^2c = c$;

y, según 1.2.3, lo anterior es igual a

$$[<c , wp(S,R)> , <N(wp(S,R)) , wp(S,R)>] =$$

$$= [<c , wp(S,R)> , 0] = <c , wp(S,R)>.$$

Pero si el elemento $c \in E_{WSR}$, entonces $c \to wp(S,R)$ y $<c , wp(S , R)> = c$, con lo que se obtiene que $(N_{WSR})^2(c) = c$.

d) Y, por último, si calculamos $[N_{WSR}c_1 , c_2]$ para dos elementos c_1, $c_2 \in E_{WSR}$ tales que existe un m: $c_1 \to c_2$, tendremos $[N_{WSR}c_1 , c_2] = [<Nc_1 , wp(S,R)> , c_2] =$
$= <[Nc_1 , c_2] , [wp(S,R) , c_2]>$;
pero si $c_1 \to c_2$, por las propiedades de C, $[Nc_1 , c_2] =$
$= 1_C$ y, por otro lado, por ser $c_2 \in E_{WSR}$, $[wp(S,R) , c_2] = wp(S,R)$; por tanto, $[N_{WSR}c_1 , c_2] =$
$= < 1_C , wp(S,R)> = wp(S,R) = 1_{EWSR}$.

A la inversa, si consideramos que $[N_{WSR}c_1 , c] = 1_{EWSR}$; podemos escribir $[<Nc_1 , wp(S,R)> , c_2] = wp(S,R)$
pero $[<Nc_1 , wp(S,R)> , c_2] =$
$= <[Nc_1 , c_2] , [wp(S,R) , c_2]>$; y como
$c_2 \in E_{WSR}$, $[wp(S,R) , c_2] = wp(S,R)$, obtenemos
$<[Nc_1 , c_2] , wp(S,R)> = wp(S,R)$.
Si calculamos el producto $<c_1 , wp(S,R)>$ a partir de la expresión anterior, se obtiene
$<c_1 , wp(S,R)> = <c_1 , <[Nc_1 , c_2] , wp(S,R)>> =$
$= <[Nc_1 , c_2] , <c_1 , wp(S,R)>> = <[Nc_1 , c_2] , c_1 >$
(pues $c_1 \to wp(S,R)$) $= [<Nc_1 , c_1> , < c_2 , c_1>] =$
$= <c_2 , c_1>$;

Ahora bien, $\langle c_1 , wp(S,R)\rangle = c_1$ y hemos obtenido $c_1 = \langle c_2 , c_1\rangle$ lo que significa que en E_{WSR} existe una flecha $c_1 \rightarrow c_2$.

Así pues, el conjunto E_{WSR} con el funtor N_{WSR} tiene estructura de N-categoría.

Es decir, según hemos visto en el apartado anterior, el conjunto E_{WSR} (cuyo objeto terminal es wp(S,R)) es el conjunto de todas las posibles precondiciones de S {R} y, por tanto, dicho conjunto de precondiciones del fragmento anotado de programa tiene estructura de N-categoría.

4. EL FUNTOR WP.

4.1. PROPIEDADES DEL OPERADOR WP EN LA N-CATEGORIA.

Independientemente de que wp(S,R) sea un objeto de la N-categoría E_{WSR} (y su elemento definidor), es un objeto de la N-categoría C de las proposiciones, de la que también son objetos las postcondiciones (en la notación que estamos utilizando, R).

Se pueden establecer, por tanto, relaciones entre los diversos operadores wp definidos para un programa S dado y diferentes postcondiciones.
En efecto, de la propia definición de wp(S,R), se deduce que

4.1.1. wp(S,0) = 0 , donde 0 es el objeto inicial de la categoría C (una proposición contradictoria) , pues ningún estado posible de conclusión del programa S satisface una proposición contradictoria;

4.1.2. wp(S,1) representa el conjunto de todos los estados tales que si S comienza su ejecución en uno cualquiera de ellos, concluye. Como, en general, es concebible que haya estados tales que tomados como comienzo de la ejecución de S, dicha ejecución no concluya, wp(S,1) no tiene por qué ser 1.

4.1.3. $\langle wp(S,R_1) , wp(S,R_2)\rangle = wp(S,\langle R_1 , R_2\rangle)$, para dos postcondiciones cualesquiera.

4.1.4. $[wp(S,R_1) , wp(S,R_2)] = wp(S,[R_1 , R_2])$.
En general, la igualdad anterior no tiene por qué cumplirse (solo se cumple la implicación hacia la derecha) si el programa S es no-determinista. Los lenguajes usuales de programación, que aquí nos interesan, son deterministas en sí, y las implementaciones de los algoritmos en ellos suelen serlo también (no lo son, por ejemplo, si se incluye una generación de números aleatorios, se opera con registros cuyo contenido no se ha controlado previamente, etc); por tanto podemos, en la mayoría de los casos, tomar la equivalencia en vez de la implicación, en la expresión del comienzo de este párrafo.

4.1.5. Si $R_1 \rightarrow R_2$ (con la relación de orden implícita en la N-categoría C), entonces $wp(S,R_1) \rightarrow wp(S,R_2)$. Esta expresión es sumamente interesante porque como las precondiciones de S{R} están relacionadas mediante $\rightarrow$ con wp(S,R), pueden extenderse las relaciones de implicación desde las precondiciones a las postcondiciones (veremos otras propiedades de relación entre pre y postcondiciones más adelante).

4.1.6. Si consideramos todas las posibles precondiciones (todas las proposiciones), el conjunto de las que cumplen: "si S comienza en un estado que cumple esa condición, concluye en otro s tal que s(R) = T (verdad)" es, precisamente, wp(S,R). Si S comienza en un estado que cumple otra condición distinta de las wp(S,R), pueden suceder tres cosas: que S no termine (en un tiempo finito), que comcluya en un estado S tal que s(R) = F (no se cumple R; es decir, se cumple NR; ó sea, s(NR) = T) ó que s(R) quede indefinido. El conjunto de estos tres conjuntos de condiciones se puede considerar como "complementario" de wp(S,R): en la N-catetgoría C es el Nwp(S,R).

posibles precondiciones:

- wp(S,R) : S concluye en s | s(R) = T
- wp(S,NR): " " " " | s(R) = F
- I : " " " " | s(R) = indef.

} Nwp(S,R)

- NT : S no termina (en tiempo finito)

Es claro, por tanto, que wp(S,NR) → Nwp(S,R), pero no coinciden. Solamente si los conjuntos que hemos llamado I y NT son vacíos, se podrá escribir wp(S,NR) = Nwp(S,R).

4.2. WP, FUNTOR ENTRE N-CATEGORIAS.

La relación entre los distintos conjuntos de condiciones en un programa con aserciones, se puede ampliar a través de las relaciones que el operador wp introduce. En efecto, las precondiciones de un módulo de programa son postcondiciones para el módulo anterior, y la relación entre estos conjuntos se establece a través del operador wp.

Sea un programa con aserciones

$$\{Q\}\ S_1\ \{R_1\}\ S_2\ \{R_2\}\ S_3\ \{R_3\}..... \qquad \textbf{[F25]}$$

Si centramos nuestra atención en el módulo S_2, encontramos su

postcondición R_2 como precondición del módulo siguiente S_3 : podemos considerar, por tanto, el conjunto E_{WS3R3} correspondiente; es claro, por la definición 4.1.1.1. que, $R_2 \rightarrow wp(S_3,R_3)$ y que $R_2 \in E_{WS3R3}$.

(En el caso de la última postcondición Rn, que no es precondición de ningún otro fragmento de programa, podemos construir también otro conjunto con la misma estructura que los E_{WSR} haciendo

$$E_{Rn} = \{c \in C \mid c \rightarrow R_n) \; ; \; N_{Rn}(c) = \langle N_C , R_n \rangle$$

que tiene, asímismo, estructura de N-categoría).

Pero, por otro lado, $R_1 \rightarrow wp(S_2,R_2)$ y, también, $R_1 \in E_{WS2R2}$, en el módulo de programa S_2.

Como sucede que $R_2 \rightarrow wp(S_3,R_3)$, según 4.1.2.5.

$$wp(S_2,R_2) \rightarrow wp(S_2,wp(S_3,R_3))$$

y, por tanto, $R_1 \rightarrow wp(S_2,wp(S_3,R_3))$ para cualquier R_1.

Por otro lado, por la propia definición de wp, $wp(S_2,wp(S,R_3))$ representa los estados que garantizan que el comienzo de la ejecución de S_2 en uno cualquiera de ellos hace que la conclusión de S_3 (pasando por algún estado intermedio que no es de interés en este momento) se efectúe en un estado que satisface R_3: es, por tanto, el objeto terminal de la

categoría $E_{WS2WS3R3}$.

Así pués, podemos considerar para el fragmento de programa S_2 un conjunto de N-categorías correspondientes a las posibles postcondiciones: E_{WS2R2}, pero todas ellas cumplen $E_{WS2R2} \subset \subset E_{WS2WS3R3}$.

Si nuestro interés es la verificación de programas, dispondremos ya de la expresión [F25] a verificar y las relaciones mencionadas, junto con las propiedades del operador wp y las derivadas de la estructura N-categorial de los conjuntos de pre y postcondiciones, deben ser suficientes para realizar dicha validación.

Si, por el contrario, nuestra tarea es la programación (la construcción de programas a partir de sus postcondiciones y de las precondiciones que los estados-datos de comienzo deben cumplir) nos será interesante seguir considerando no solo las posibles pre-postcondiciones intermedias, R_i, que aparecen en [F25], sino los conjuntos de todas las posibles pre y post-condiciones; es decir los E_{WSiRj}.

Tal como hemos visto, en general podemos considerar no solo la

postcondición R_2 de S_2, sino el conjunto de todas sus postcondiciones (que son las posibles precondiciones de S_3 para obtener R_3): $wp(S_3,R_3)$. Entonces, mediante el operador wp, actuando como funtor covariante entre N-categorías obtenemos la imagen correspondiente $wp(S_2,wp(S_3,R_3))$ a partir de la cual obtenemos el conjunto de todas las posibles precondiciones de S_2 : $E_{WS2WS3R3}$.

Estas relaciones entre los elementos terminales de las correspondientes N-categorías de pre y postcondiciones se pueden prolongar a lo largo de toda la cadena de fragmentos de programa y aserciones.

Según vemos, el funtor covariante wp entre estas N-categorías realiza una translación total de las estructuras de los conjuntos al hacer corresponder:

> -al elemento terminal de una N-categoría de postcondiciones, el elemento terminal de la N-categoría de las precondiciones; esto es así, aunque no sucede lo mismo, como sabemos, con el conjunto C de todas las posibles condiciones (proposiciones), pués en C <u>no sucede</u> que $wp(S_i,1) = 1$, según 4.1.2.)

-al elemento inicial de las postcondiciones, el elemento inicial de las precondiciones (en ambos casos es 0 y se corresponden según 4.1.1.) ;

-al producto de dos elementos, su producto en la N-categoría codominio del funtor (ver 4.1.3).

-al coproducto el coproducto (en los casos de programas deterministas, según se comenta en 4.1.4);

5. LA N-CATEGORIA DE LAS GUARDAS

5.1. EL CONJUNTO INC(X) ES UNA CATEGORIA PREORDEN

5.1.1. Naturalmente, el conjunto de subconjuntos de uno dado X, con las funciones de inclusion correspondientes, forma una categoría (como subconjunto de la categoría de las funciones parciales $Pf_n(X,X)$), donde los objetos son los subconjuntos del conjunto X y los morfismos las propias funciones de inclusión; pero se puede definir otra categoría en el conjunto de las funciones de inclusión tomando como objetos las propias funciones de inclusión (elementos de INC(X)), y como flechas los morfismos definidos por una relación de diferencia, como se define a continuación.

A partir de la operación suma definida en las multifunciones (Mfn) (ver 1.3.1.2.) se pueden definir unos morfismos en el conjunto de las funciones de inclusión del siguiente modo: dadas dos funciones de inclusión (g_1, $g_2 \in INC(X)$) de un conjunto cualquiera X ($g_i \in Mfn(X,X)$) se dice que $g_1 \overset{f}{\rightarrow} g_2$ si y solo si existe al menos una función $g_d \in INC(X) \subset Mfn(X,X)$

tal que $g_2 = g_1 + g_d$ [F26]

Cada una de estas funciones de inclusión ($g_1, g_2, g_d \in INC(X)$) corresponden a un subconjunto de X; es decir $g_1 \equiv inc_A$, $g_2 \equiv inc_B$, $g_d \equiv inc_C$ para ciertos A, B, C $\subset$ X.

5.1.2. *Se puede comprobar que el conjunto INC(X) y el conjunto* de morfismos definidos forman una categoria (ver 1.1.1.1.) estando relacionados mediante las correspondientes funciones dom y cod: dado el morfismo f: $g_1 \rightarrow g_2$ es

$$g_1 = \text{dom } f \quad y \quad g_2 = \text{cod } f,$$

pues la pareja g_1, g_2, de [F26] define un único morfismo f, sea cual sea la g_d elegida en la suma. Además, se pueden definir las dos operaciones siguientes:

$b_{1)}$ identidad;

asigna a cada función de inclusión g el morfismo idg: g $\rightarrow$ g definido mediante la expresión $g = g + g_o$, donde g_o es el morfismo inc_Φ definido en 1.3.4.3.1.; en efecto, para toda función g_1, siempre se cumple $g_1 = g_1 + g_o$, pues en la suma $\Sigma_{0,1}\ g_i(x) = U_{0,1}\ g_i(x) =$ $= \{y \in X \mid y \in g_{iX} \mid y \in g_1(x) \text{ ò } y \in \Phi\}$ es claro que $y \notin \Phi$ y por tanto, necesariamente, $y \in g_1(x)$;

b_2) composición ($\cdot$);

asigna a cada pareja de morfismos f_1 y f_2 tales que dom f_2 = cod f_1 un único morfismo $f_2 \cdot f_1$: dom $f_1 \to$ cod f_2 ; efectivamente si son f_1: $g1 \to g_2$ y f_2: $g_2 \to g_3$ sucederá que $\exists\ g_{d1}$ y g_{d2} tales que $g_2 = g_1 + g_{d1}$ y $g_3 = g_2 + g_{d2}$ de modo que se puede escribir

$g_3 = (g_1 + g_{d1}) + g_{d2}$, pero esta suma de multifunciones es asociativa, pues en

$$((g_1 + g_{d1}) + g_{d2})\ (x) = (g_1 + g_{d1})\ (x)\ \cup\ g_{d2}\ (X) =$$

$$= \{\ y \in X \mid y \in (g_1 + g_{d1})(x) \text{ ó bién } y \in g_{d2}\ (x)\ \}$$

pero que $y \in (g_1 + g_{d1})(x)$ quiere decir que

$y \in g_1(x) \cup g_{d1}(x) = \{\ y_2 \in X \mid y_2 \in g_1(x)$ ó $y_2 \in g_{d1}(x)\}$;

en definitiva,

$$((g_1 + g_{d1}) + g_{d2})(x) = \{y \in X \mid y \in g_1\ (x)$$

ó $y \in g_{d1}(X)$ ó $y \in g_{d2}(x)$, al menos a uno de ellos $\}$;

por tanto,

$$(g_1 + g_{d1}) + g_{d2} = \qquad \text{[F27]}$$

$$= g_1 + (g_{d1} + g_{d2}) = g_1 + g_{d1} + g_{d2};$$

es decir,

$$g_3 = g_1 + (g_{d1} + g_{d2}) \cdot$$

Pero si g_{d1}, g_{d2} ϵ INC(X), entonces $g_{d1}(x) = inc_A(x)$ y $g_{d2}(x) = incB(x)$ para algunos A, B $\subset$ X,

y, al calcular la suma,

$(g_{d1} + g_{d2})(x) = \{ y \epsilon X \mid y \epsilon g_{d1}(x)$ ó $y \epsilon g_{d2}(x) =$

$= \{ y \epsilon X \mid y \epsilon A$ ó $y \epsilon B \} = \{ y \epsilon X \mid y \epsilon A \cup B \}$

por la propia definición de A U B; es decir,

$(g_{d1} + g_{d2})(x) = inc_{AUB}$ y es, por tanto, una función de inclusión de INC(X).

Así pues, $g_3 = g_1 + g_{AUB}$ y, en consecuencia, existe una función de inclusión que cumple la condición de **[F26]** y existe, por tanto, un morfismo $f_3 : g_1 \rightarrow g_3$ que es el resultado de la composición de f_1 y f_2, es decir, $f_3 = f_2 \cdot f_1$.

Por último, se puede ver que las dos operaciones definidas cumplen los dos axiomas **[F2]:**

Ax1 : Dadas cuatro funciones de inclusión

$g_i \epsilon$ INC(X) $\subset$ Mfn(X,X) i= 1,...,4 y tres flechas que las relacionan

$$g_1 \xrightarrow{f1} g_2 \xrightarrow{f2} g_3 \xrightarrow{f3} g_4$$

se cumple que

$f_3 \cdot (f_2 \cdot f_1) = (f_3 \cdot f_2) \cdot f_1$, pues si

$f_1 : g_1 \to g_2$ entonces $g_2 = g_1 + g_{d1}$

$f_2 : g_2 \to g_3$ " $g_3 = g_2 + g_{d2}$

$f_3 : g_3 \to g_4$ " $g_4 = g_3 + g_{d3}$

y podemos asegurar que por $f_2 \cdot f_1$ se cumple

$f_2 \cdot f_1 : g_1 \to g_3$ de tal modo que

(según [F27])

$$g_3 = g_2 + g_{AUB} = g_1 + (g_{d1} + g_{d2})$$

siendo A, B $\subset$ X respectivamente, los conjuntos cuyas funciones de inclusión son g_{d1} y g_{d2}; si ahora componemos f_3 con la flecha compuesta de f_2 y f_1

$$f_3 \cdot (f_2 \cdot f_1) : g_1 \to g_4$$

que, por la misma razón cumple

$$g_4 = g_1 + g_{(AUB)UC} = g_1 + ((g_{d1} + g_{d2}) + g_{d3}),$$

donde $C \subset X$ es tal que $g_{d3}(x) = inc_C(x)$.

Del mismo modo, si razonamos con la composición de f_1 con el resultado de componer f_2 con f_3, para obtener $(f_3 \cdot f_2) \cdot f_1$, se cumplirá:

$$(f_3 \cdot f_2) \cdot f_1 : g_1 \to g_4$$

que, de acuerdo con la secuencia de composición aplicada ahora, permite escribir

$$g_4 = g_1 + g_{AU(BUC)} = g_1 + (g_{d1} + (g_{d2} + g_{d3})).$$

Como las funciones g_{di} son multifunciones y su suma es

asociativa, según se ha demostrado, se cumple que

$$(g_{d1} + g_{d2}) + g_{d3} = g_{d1} + (g_{d2} + g_{d3}),$$

lo que identifica los dos resultados obtenidos para, respectivamente, $f_3 \cdot (f_2 \cdot f_1)$ y $(f_3 \cdot f_2) \cdot f_1$.

Ax2: Dadas tres funciones de inclusión,

$g_i \in INC(X) \subset Mf_n(X , X)$ i=1,2,3 y dos morfismos que las relacionan

$$g_1 \overset{f_1}{\rightarrow} g_2 \overset{f_2}{\rightarrow} g_3$$

la composición con el morfismo identidad de g_2 (idg_2) cumple que

$$idg_2 \cdot f_1 = f_1 \quad y \quad f_2 \cdot idg_2 = f_2$$

En efecto, se puede escribir que

si existe $f_1: g_1 \rightarrow g_2$ entonces $g_2 = g_1 + g_{d1}$

$idg_2: g_2 \rightarrow g_2$ entonces $g_2 = g_2 + g_0$

$f_2: g_2 \rightarrow g_3$ entonces $g_3 = g_2 + g_{d2}$

y, según la propiedad **[F27]** de la composición de morfismos en INC(X),

$idg_2 \cdot f_1: g_1 \rightarrow g_2$ tal que

$g_2 = g_1 + g_{AUC}$ donde A es tal que

$inc_A = g_{d1}$ y C el correspondiente conjunto tal que

$inc_C = inc_\Phi = gd_0$; por tanto, AUC = AUΦ = A, por lo que

$g_{AUC} = g_A = g_{d1}$ y resulta (en $idg_2 \cdot f_1$) $g_2 = g_1 + g_{d1}$

igual que en f_1.

El mismo razonamiento se puede hacer con $f_2 \cdot id_{g_2}$, pues $g_{CUB} = g_{\Phi UB} = g_B = g_{d2}$, con lo que resulta $f_2 \cdot id_{g_2} = f_2$.

5.1.3. CATEGORIA PREORDEN

En efecto, la categoría INC(X) es preorden, pues en la definición **[F26]** del morfismo que relaciona dos funciones de inclusión, se afirma la existencia de dicha flecha, cualquiera que sea la función f_d elegida (y pueden existir varias). Por tanto, si existe al menos una f_d que cumple **[F26]** se define el morfismo; si no hay ninguna f_d que satisfaga dicha relación, no hay flecha entre g_1 y g_2.

5.2. EL CONJUNTO INC(X) ES UNA N-CATEGORIA.

Veamos que el conjunto INC(X) es, además, una N-Categoría. En efecto, comprobemos que (según **[F11]**) cumple:

5.2.1. INC(X) tiene un objeto terminal. Este es el morfismo $inc_X = id_X$.

En efecto, un objeto cualquiera $g \in INC(X)$ supone un conjunto

$A \subset X$ tal que $g = inc_A$. A partir de A, siempre se puede encontrar otro conjunto diferencia B, formado por los elementos de X que no pertenecen a A. Es evidente que, en estas condiciones, $x \in X : x \in AUB$ y, por tanto, $x \in g_{AUB}$; asi pues, $inc_X = inc_A + inc_B$, según la propiedad **[F26]**. Es decir, $id_X = g + g_d$ siendo $g_d = inc_B$. Por tanto, existe un g_d que, sumado a g, da id_X, o, lo que es lo mismo, existe un $f: g \to id_X$.

La unicidad de f viene asegurada por el hecho de ser INC(X) una categoría preorden (ver 4.1.1.3).

5.2.2. INC(X) tiene coproductos finitos [-,-].

En efecto, podemos definir el coproducto de dos funciones de inclusión $g_1, g_2 \in INC(X)$ como

$$[g_1 , g_2] = \Sigma_{1,2}\, g_i$$

Puesto que, según la definición, $[g_1 , g_2] = g_1 + g_2$, existen sendos morfismos

$f_1: g_1 \to [g_1 , g_2]$ y $f_2: g_2 \to [g_1 , g_2]$. Por otro lado, si tenemos que existe otra función g_3 tal que $f_{31}: g_1 \to g_3$ y $f_{32}: g_2 \to g_3$ esto significa que

$g_3 = g_1 + g_{d13}$ y $g_3 = g_2 + g_{d23}$, donde

$g_{d13}, g_{d23} \in INC(X)$; pero esto quiere decir que

$\forall x \in X$, $g_3(x) = \{ y \in X \mid y \in g_1(x)$ ó $y \in g_{d13}(x)$ ó ambos$\}$ y

también que

$g_3(x) = \{ y \in X \mid y \in g_2(x)$ ó $y \in g_{d23}(x)$ ó ambos$\}$;

reuniendo ambas afirmaciones (que se cumplen simultáneamente), podemos asegurar que

$g_3(x) = \{ y \in X \mid y \in g_1(x)$ ó $y \in g_2(x)$ ó $y \in g_{d23}(x)$ ó varios$\} =$

$= \{ y \in X \mid y \in g_1(x) \cup g_2(x)$ ó $y \in g_{d13}(x) \cup$

$\cup g_{d23}(x)$ ó ambos $\} = (g_1+g_2)(x) + (g_{d13} + g_{d23})(x)$;

es decir, $g_3 = (g_1 + g_2) + g_{d13Ud23}$, según [F27], lo que indica que existe un morfismo f_3 tal que

$$f_3: [g_1 , g_2] \rightarrow g_3$$

5.2.3. En INC(X) existe un funtor contravariante (N). En efecto, es el morfismo que hace corresponder a cada función de inclusión $g_1 \in$ INC(X) otra que sumada (suma de multifunciones) con ella dé el objeto terminal (esto es, el morfismo id_X) y que convierte cada morfismo f entre dos funciones de inclusión g_1 y g_2 tal que $f: g_1 \rightarrow g_2$, en otro Nf en sentido opuesto, es decir: $Nf: Ng_2 \rightarrow Ng_1$.

Una $g_1 \in$ INC(X) viene definida por un subconjunto $A \subset X$ tal que $g_1(x) = inc_A(x)$ $\forall x \in X$, como sabemos. Es facil concebir el conjunto B de elementos de X que hace que $inc_B + inc_A = id_X$. La función de inclusión $g_2 \in$ INC(X) definida por ese conjunto B, $g_2(x) = inc_B(x)$ es precisamente la

imagen por N de la función de inclusión $g_1 : g_2 = Ng_1$.
Por otro lado, de f_1: $g_1 \rightarrow g_2$ se deduce $g_2 = g_1 + g_d$.
Puesto que acabamos de definir que $id_X = g_2 + Ng_2$ y, también, $id_X = = g_1 + Ng_1$, podemos asegurar que
$id_X = g_2 + Ng_2 = g_1 + Ng1$ (es decir que $\forall$ y ϵ X será y ϵ $g_2(x)$ ó y ϵ $Ng_2(x)$ y también y ϵ $g_1(x)$ ó y ϵ $Ng_1(x)$).
Pero, a partir de la definición de la flecha f_1, $g_1 + g_d + Ng_2 = g_1 + Ng_1$; es decir, que
$\forall$ y ϵ X | y ϵ $(g_1 + g_d + Ng_2)(x)$ sucede que y ϵ $(g_1 + Ng_1)(x)$; o, lo que es lo mismo, $\forall$ y ϵ X | y ϵ $g_1(x)$ ó y ϵ $g_d(x)$ ó y ϵ $Ng_2(x)$ ó a varios, sucede que y ϵ $g_{1(}x)$ o y ϵ $Ng_1(x)$ o ambos; por tanto,
$\forall$ y ϵ X | y ϵ $g_d(x)$ o y ϵ $Ng_2(x)$ o y ϵ a ambos, sucede que y ϵ $Ng_1(x)$. Es decir, $g_d + Ng_2 = Ng_1$ y, por tanto, existe un morfismo f_2: $Ng_2 \rightarrow Ng_1$; este morfismo es el que definimos como imagen del f_1 respecto del funtor contravariante N.

Además, es facil razonar que el morfismo N^2 es naturalmente equivalente al morfismo identidad en INC(X); en efecto, la propia definición del funtor N establece que $g + Ng = id_x$ para cada función de inclusión g ϵ INC(X) y su correspondiente imagen Ng.
Pero la misma expresión leida tomando Ng como la función de

inclusión que estamos considerando, indica que g es la imagen por N de Ng; es decir que $NNg = g$, para toda $g \in INC(X)$.

5.2.4. Existe en INC(X) un morfismo $f: g_1 \to g_2$ si y solo si $[Ng_1 , g_2] = id_X$, $\forall\ g_1, g_2 \in INC(X)$.

Como sabemos, si existe f entonces $g_2 = g_1 + g_d$.

Si consideramos $[Ng_1 , g_2] = Ng_1 + g_2$ y sustituimos el valor de g_2 anterior, obtenemos $[Ng_1 , g_2] = Ng_1 + g_1 + g_d$. Acabamos de establecer que $Ng_1 + g_1 = id_X$ y, por tanto, $[Ng_1, g_2] = id_X + g_d$. Pero sucede que

$(id_X + g_d)(x) = \{ y \in X \mid y \in id_X$ ó $y \in g_d(x)$ o ambos $\}$ lo que equivale a decir $\{ y \in X \mid y \in id_X \}$, pues $\forall\ y \in X$, $y \in id_X$ independientemente de que $y \in g_d(x)$ o no: por tanto, $[Ng_1 , g_2] = id_X$.

En sentido contrario, sabemos que si para dos funciones de inclusión cualesquiera $g_1, g_2 \in INC(X)$ se cumple que $[Ng_1 , g_2] = id_X$, se cumple también $Ng_1 + g_2 = id_X$. Además, por otro lado, también $Ng_1 + g_1 = id_X$, por la propia definición del funtor N.

Así pues, podemos asegurar que $\forall\ y \in X$: $y \in Ng_1(x)$ ó $y \in g_1(x)$ o ambos y también que $y \in Ng_1 (x)$ ó $g_2(x)$ o ambos; como ningún elemento puede pertenecer simultaneamente a $g_1(x)$ y a $Ng_1(x)$ (por la definición de $Ng_1(x)$) pero si a $Ng_1(x)$ y a

$g_2(x)$, es claro que en $g_2(x)$ existen al menos los elementos de $g_1(x)$ pero pueden estar alguno mas. Por tanto podemos escribir

$$inc_{g2}(x) = inc_{g1} + inc_d(x),$$

donde $inc_d(x)$ puede ser vacio; es decir, $g_2 = g_1 + g_d$ que nos asegura que existe un morfismo f: $g_1 \rightarrow g_2$.

5.3. PROPIEDADES DE LA N-CATEGORIA INC(X)

El conjunto INC(X) también cumple, obviamente, las propiedades indicadas en 1.2.2. para cualquier N-categoría.

5.3.1. El objeto inicial es, en este caso, la función de inclusión g_0 ya utilizada, definida a partir del conjunto vacio; es decir, $g_0 = inc_\Phi$. En efecto se cumple que $Ng_I = g_0$, lo que significa $Nid_X = inc_\Phi$, pues $id_X = inc_X$ y es evidente que el conjunto formado por los puntos que, perteneciendo a X, no pertenecen a X (ver 3.1.2.3.) es el vacio.

5.3.2. EXISTEN PRODUCTOS FINITOS.

En efecto, si definimos

$$\langle g_1(x), g_2(x)\rangle = g_2(g_1(x)),\ g_1,\ g_2 \in INC(X),$$

se cumple que $\langle g_1 , g_2\rangle = N[Ng_1 , Ng_2]$, como debe suceder en toda N-categoría.

En efecto, se cumple que

$$[Ng_1(x) , Ng_2(x)] = Ng_1(x) + Ng_2(x) =$$

$$= \{ y \in X \mid y \in Ng_1(x) \text{ ó } y \in Ng_2(x) \text{ o ambos}\};$$

pero, por otro lado,

$g_2(g_1(x)) = \{ y \in X \mid y \in g_1(x)$ y también $y \in g_2(x)\}$.

5.4. PLANTEAMIENTO ABSTRACTO: LA N-CATEGORIA G(X).

Las propiedades de 1.3.1.3.5 que cumplen las funciones de inclusión (funciones de inclusión que realizan el papel de "guardas" en las estructuras de programa), sugieren definir un conjunto abstracto (que llamaremos "guardas", por lo anterior) en las categorías parcialmente aditivas. Las funciones de guarda que aparecen en los programas (en un enfoque de semántica parcialmente aditiva) cumplen la definición que, para el conjunto G(X), daremos a continuación, pero este conjunto no se limita solamente al caso de las guardas (como funciones de inclusión): estas funciones son un caso particular del conjunto G(X) que se puede definir en cualquier categoría parcialmente aditiva.

5.4.1 DEFINICION DEL CONJUNTO G(X)

Dado X, objeto cualquiera de una categoría parcialmente

aditiva C, se define G(X) (ver [MAN86] pg. 89) como el subconjunto de C(X,X) formado por todos los morfismos f para los que existe otro f' tal que

$$f + f' \text{ existe y es tal que } f + f' = id_X$$
$$f \cdot f' = f' \cdot f = 0 \qquad \textbf{[F28]}$$

donde + es la suma de la categoría parcialmente aditiva, · la composición de funciones y 0, 1, respectivamente los elementos inicial y terminal (ver 1.1.4.4.).

5.4.2. Propiedades inmediatas en G(X).

De la definición **[F28]** del morfismo complementario de otro en la categoría parcialmente aditiva C(X,X) se deducen algunas propiedades inmediatas sumamente interesantes:

a) El elemento $f' \in G(X)$ complementario de $f \in G(X)$ es único, pues si suponemos que existe otro $g \in G(X)$ que cumple las propiedades **[F28]**:

$f + g = 1$ y $f \cdot g = g \cdot f = 0$,

podemos calcular

$g = g \cdot 1 = g \cdot (f + f') = g \cdot f + g \cdot f' =$

$= 0 + g \cdot f' = g \cdot f'$

y, por tanto,

$f' = 1 \cdot f' = (f + g) \cdot f' = f \cdot f' + g \cdot f' =$

$= 0 + g \cdot f' = g \cdot f' = g$

b) Además, $f'' = f$ **[F29]**

como se puede comprobar si en las ecuaciones de definición de f' **[F28]**, nos fijamos en el elemento f' y constatamos que f cumple las condiciones necesarias para ser su complementario.

c) También se cumple que

$0' = 1$ y $1' = 0$ **[F30]**

como se comprueba sin más que escribir las igualdades

$$0 + 1 = 1$$

$$0 \cdot 1 = 0 = 1 \cdot 0$$

5.4.3. G(X) TIENE ESTRUCTURA DE CATEGORIA PREORDEN

Se define un morfismo en el conjunto C(X,X), del siguiente modo:

$f \rightarrow g$ si existe un h tal que $g = f + h$ **[F31]**

donde f, g, h ϵ C(X,X) y la suma (+) es la suma de la categoría parcialmente aditiva C (X,X) (ver 1.1.4.4).

El conjunto G(X) es una categoría preorden con la flecha que acaba de ser definida en [F31].
En efecto, se pueden definir las dos operaciones básicas (ver 1.1.1.1.):

b_1) identidad:

a cada elemento f de G(X) se le hace corresponder el morfismo id_f: f → f definido por

$$f = f + 0$$

b_2) Composición:
dados dos flechas f → g y g → h que corresponden a $g = f + d_1$ y $h = g + d_2$, se define f → h por $h = g + d_2 = f + d_1 + d_2 = f + d_3$ y, obviamente, si $d_1, d_2 \in G(X)$, entonces $d_3 \in G(X)$.

Por otro lado, ambas operaciones cumplen los axiomas definidos en [F2]:
Si tenemos $g = f + d_1$, $h = g + d_2$ e $i = h + d_3$, realizando la sustitución de valores correspondiente, llegamos a

$h = g + d_2 = (f + d_1) + d_2 = f + (d_1 + d_2)$,

pues la suma de 1.1.4.4. es asociativa;

y del mismo modo

$i = h + d_3 = f + ((d_1 + d_2) + d_3)$.

Pero si hacemos

$i = h + d_3 = (g + d_2) + d_3 = g + (d_2 + d_3)$

y después

$i = g + (d_2 + d_3) = (f + d_1) + (d_2 + d_3) =$

$= f + (d_1 + (d_2 + d_3))$,

obtenemos el mismo resultado, ya que

$d_1 + (d_2 + d_3) = (d_1 + d_2) + d_3$

si d_1, d_2, $d_3 \in C(X,X)$, pues la suma es asociativa.

Por tanto, podemos asegurar que si llamamos

$$f \overset{m_1}{\rightarrow} g \overset{m_2}{\rightarrow} h \overset{m_3}{\rightarrow} i$$

se cumplirá

$$m_3 \cdot (m_2 \cdot m_1) = (m_3 \cdot m_2) \cdot m_1.$$

Por otro lado, puesto que $g \overset{idg}{\rightarrow} g$ significa que

$g = g + 0$, se puede escribir

$m_1 = idg \cdot m_1$ y $m_2 = m_2 \cdot idg$,

puesto que

$g = g + 0 = (f + d_1) + 0 = f + (d_1 + 0) = f + d_1$

y también

$h = g + d_2 = (g + 0) + d_2 = g + (0 + d_2) = g + d_2$.

Por último, la existencia de la flecha $f \rightarrow g$ viene dada (según la definición [F31]) por el hecho de existir un h que cumpla

$$g = f + h.$$

Si existe este h (ó si existen varios, lo mismo dá) se define la flecha; en caso contrario, no existe flecha: por tanto, si hay morfismo $f \rightarrow g$, es único y la categoría G(X) es preorden.

5.4.4. G(X) es una N - categoría.

Ante todo, veamos que la condición de definición de la flecha en G(X):

$f \rightarrow g$ <==> $g = f + h$ (ver [F31]) es equivalente a

$f \rightarrow g$ <==> $g \cdot f = f$. [F32]

En efecto, si $f = g \cdot f'$ tenemos

$g = (f + f') \cdot g = f \cdot g + f' \cdot g = f + f' \cdot g = f + h$, haciendo $h = f' \cdot g$.

Por el contrario, si $g = f + h$,

se puede escribir

$g \cdot g' = (f + h) \cdot g' = f \cdot g' + h \cdot g'$

y como $g \cdot g' = 0$,

se obtiene (según 1.1.4.4.2.) $f \cdot g' = 0$ y $h \cdot g' = 0$;

al calcular

$f = f \cdot (g + g') = f \cdot g + f \cdot g' = f \cdot g + 0$,

según el resultado anterior;

es decir $f = f \cdot g$, lo que demuestra la equivalencia de ambas caracterizaciones de la existencia de flecha.

Además, hemos visto que

$f \cdot g = f \iff f \cdot g' = 0$

y, de la equivalencia anterior, **[F32]**, se deduce

$$f \to g \iff f \cdot g' = 0 \qquad \textbf{[F33]}$$

Veamos ahora que se cumplen las condiciones exigidas en **[F11]**. Tomemos como funtor contravariante $N: G(X) \to G(X)$ el morfismo que hace corresponder a cada elemento $f \in G(X)$ su elemento $f' \in G(X)$ definido en **[F28]**. La correspondencia definida por el morfismo N entre flechas de la categoría G(X) es

$Nm_1 = m_2$ donde $m_1: f \to g$ y $m_2: Ng \to Nf$.

Este morfismo N es, en efecto, un funtor, pues cumple las propiedades exigidas en **[F6]**:

a) si consideramos el morfismo identidad, idf, de un objeto cualquiera de la categoría $f \in G(X)$, $idf: f \to f$, el correspondiente morfismo imagen es $Nidf = N(f \to f) =$
$= Nf \to Nf = idNf$;

b) respecto de la composición de flechas, se vé que
$N(m_1 \cdot m_2) = N((f \to g) \cdot (g \to h)) = N(f \to h) =$
$= Nh \to Nf$,
para dos morfismos cualesquiera de la categoría
$m_1: f \to g$ y $m_2: g \to h$; $f, g, h \in G(X)$;
pero, por otro lado,
$Nm_1 : Ng \to Nf$ y $Nm_2 : Nh \to Ng$,
y al componerlos,
$Nm_1 \cdot Nm_2 = (Ng \to Nf) \cdot (Nh \to Ng) = Nh \to Nf$,
por lo que

$$N (m_1 \cdot m2) = Nm_1 \cdot Nm_2$$

Este funtor es, además, contravariante, porque si tenemos

$$m_1: f \to g$$

(es decir, $g = f + d_1$, para algún $d_1 \in G(X)$ ó bien $g \cdot f = f$ ó $f \cdot g' = 0$, según acabamos de ver) los objetos imagen mediante el funtor son

$$Nf = f' \quad y \quad Ng = g',$$

y se cumple que

$$Nm_1: Ng \to Nf,$$

pues si calculamos

$$Ng \cdot N(Nf) \quad \text{(ver [F33])},$$

se tiene

$g' \cdot (f')' = g' \cdot f$ (según [F29]) y $g' \cdot f = 0$, puesto que $f \to g$, por hipótesis.

a) G(X) tiene como objeto terminal el 1 de la categoría parcialmente aditiva C(X,X) en la que está definido (G(X) $\subset$ C(X,X)), puesto que para cualquier elemento $f \in G(X)$, por definición (ver [F28]) existe otro (f') tal que

$1 = f + f'$ y, por tanto, $f \to 1$.

b) G(X) tiene coproductos finitos.

Definimos

$$[f , g] = f + f'g$$

y, en efecto, se cumple que

$$f \to [f , g] = f + f' \cdot g ,$$

pues $\forall f \quad \exists f'$ y, por tanto, $f' \cdot g$;

con lo que se vé que

$[f , g] = f + h_1$, tomando $h_1 = f' \cdot g$.

Además, como

$f + f' \cdot g = (f + f') \cdot (f + g) = 1 \cdot (f + g) = f + g,$

se puede escribir

$[f , g] = g + h_2$ siendo $h_2 = f$;

es decir, $g \to [f , g]$.

Por otro lado, si hay un h tal que

$$f \to h \quad y \quad g \to h,$$

se cumple

$f \cdot h = f$ y $g \cdot h = g$ (según [F32])

y si calculamos

$[f , g] \cdot h = (f + f' \cdot g) \cdot h = f \cdot h + f' \cdot g \cdot h =$

$= f + f' \cdot g = [f , g]$

con lo que podemos asegurar que

$$[f , g] \to h.$$

c) El funtor N^2 es naturalmente equivalente al morfismo identidad. En efecto, como $Np = p'$, ya tenemos comprobado (ver [F29]) que $(p')' = p$.

d) Existe en G(X) un morfismo m_1: $f \to g$ si, y solo si $[Nf , g] = 1$; $f , g, \epsilon G(X)$.

En efecto, que exista m_1: $f \to g$

significa que $f \cdot g = f$, y si calculamos

$[Nf , g] = f' + (f')' \cdot g = f' + f \cdot g = f' + f = 1.$

Por otro lado,

si [Nf , g] = 1, entonces f'+ f · g = 1,

según acabamos de ver;

pero al calcular f'· (f · g) se obtiene

f'· (f· g) = (f'· f)· g = 0 · g = 0 ,

con lo que se vé que

f'+ (f · g) = 1 y f'· (f · g) = 0

que son las condiciones de definición del complementario de f': es decir,

f · g = (f')' = f,

lo que demuestra que f → g.

5.4.5. OTRAS PROPIEDADES DE LA N-CATEGORIA G(X).

Además de las propiedades de definición de la N-categoría que acabamos de ver, en G(X) se cumplen otras propiedades:

a) Existe un elemento inicial, 0 (Ver 1.1.4.4); además, según 5.4.2, 0 = N1.

b) El producto de dos objetos f, g ∈ G(X) cumple que (ver 1.2.2.2.)

< f , g > = N [Nf , Ng] = N [f', g'] =

= N (f' + (f')' · g') = N (f' + f · g'); es decir

$< f , g >$ es el complementario de $f' + f \cdot g'$. Veamos que este objeto complementario es, precisamente, $f \cdot g$: en efecto, si calculamos

$$(f' + f \cdot g') + f \cdot g = f' + (f \cdot g' + f \cdot g) =$$

$$= f' + f \cdot (g + g') = f' + f \text{ según [F28]} = 1 ;$$

por otro lado

$$(f' + f \cdot g') \cdot (f \cdot g) = (f' \cdot f \cdot g) + (f \cdot g' \cdot f \cdot g) =$$

$$= 0 \cdot g + f \cdot 0 = 0 + 0 = 0.$$

Así pués, el producto de objetos de G(X) es la composición: $< f(x) , g(x) > = f(g(x))$.

c) Es claro que se cumple en G(X) la propiedad distributiva de los productos:

$$< [f , g] , [f , h] > = [f , g] \cdot [f , h] =$$

$$= (f + f' \cdot g) \cdot (f + f' \cdot h) =$$

$$= f \cdot f + f' \cdot g \cdot f + f \cdot f' \cdot h + f' \cdot g \cdot f' \cdot h =$$

$$= f + 0 + 0 + f' \cdot g \cdot h = f + f' \cdot g \cdot h,$$

que es, precisamente,

$$[f , g \cdot h] = [f , < g, h >].$$

d) El pseudocomplemento de f respecto de g, $f , g \in G(X)$ es

$$[Nf , g] = f' + (f')' \cdot g = f' + f \cdot g.$$

6. N-CATEGORIAS CON LA DISYUNCION EXCLUSIVA: EXPLICACION ALTERNATIVA A LA SUMA DE MULTIFUNCIONES.

6.1. TODA N-CATEGORIA ES UN "CONJUNTO DE GUARDAS".

En efecto, se puede demostrar que cualquier categoría C que cumpla las propiedades [F11] de definición de la N-categoría, tiene la estructura de un conjunto de guardas, si se define en ella una suma adecuada. Además, veremos que el preorden inducido por esa suma, coincide con la flecha de la N-categoría.

6.1.1. PROPIEDADES DE LOS PRODUCTOS DE UNA N-CATEGORIA

Sea C una N-categoría con el funtor N: C → C. Se cumple que:

6.1.1.1. [Nc , c] = 1 ; [c , Nc] = 1,

para cualquier objeto c ∈ C , siendo 1 el objeto terminal.

En efecto, ∀ c ∈ C, existe id_C : c → c y, por tanto, según d) de [F11] se cumple la primera propiedad del enunciado; del mismo modo, puesto que existe id_{NC}: Nc → Nc

se puede asegurar que [NNc , Nc] = 1

to, lo que es lo mismo, según c) de **[F11]**, [c , Nc] =1.

6.1.1.2. < Nc , c > = 0 ; < c_1 , Nc_1 > = 0, para cualquier objeto c ϵ C, siendo 0 el objeto inicial de C.

En efecto, según 1.2.2.2.

< Nc , c > = N[N Nc , Nc]

pero, según c) de **[F11]**

NNc = c , $\forall$ c ϵ C

y, por tanto,

< Nc , c > = N[c , Nc] = N1

según la propiedad anterior, y como N1 = 0 (ver 1.2.2.1), se cumple la primera propiedad del enunciado.

Por lo mismo, podemos asegurar que

< N Nc , Nc > = 0 , pues Nc ϵ C ;

y como NNc = C, se obtiene la segunda propiedad.

6.1.1.3. [1 , 1] = 1 ; [1 , c] = 1 ; [0 , c] = c para cualquier objeto c ϵ C , siendo 0, 1 los objetos inicial y final, respectivamente, de C.

En efecto, como N1 = 0 (1.2.2.1), y se cumple que existe un morfismo f: 0 $\rightarrow$ 1 , podemos escribir (ver d) de **[F11]**) que [N0 , 1] = 1 ; es decir [1 , 1] = 1.

También existe $f: 0 \to c$, $\forall\, c \in C$, por la propia definición de objeto inicial; y, por tanto, $[1 , c] = 1$. Que $[0, c]$ es c, se puede ver a partir de la definición del coproducto. En efecto, c cumple que

$$\exists\, f_1 \mid f_1: 0 \to c$$

$$\exists\, f_2 = id_c \mid id_c: c \to c$$

y, además, si hay un objeto $p \in C$ tal que

$$\exists\, f_3 \mid f_3: 0 \to p$$

$$\exists\, f_4 \mid f_4: c \to p$$

es obvio que se cumple que $c \to p$: por tanto, c es el coproducto de 0 y c.

6.1.1.4. $< 0 , 0 > = 0$; $< 0 , p > = 0$; $< 1 , p > = p$
Se obtienen a partir de las anteriores, teniendo en cuenta la propiedad 1.2.2.2. Por ejemplo,

$$< 1 , p > = N\,[N1 , Np] = N\,[0, Np] = N\,Np = p,$$

y el resto de un modo semejante.

6.1.2. DEFINICIONES.

6.1.2.1. En cualquier N-categoría se puede definir una suma en base a los productos y coproductos del siguiente modo: $a + b = < [a , b] , N< a , b > >$

6.1.2.2. Además, para establecer la relación entre los elementos de la N-categoría y los correspondientes elementos definidos en el conjunto $G(X) \subset C(X , Y)$, consideraremos como elemento equivalente de id_X el objeto terminal de la N-categoría ($id_X = 1$) ; tomaremos, también, como elemento complementario f' de uno dado f (ver **[F28]**) el elemento Nf obtenido como imagen por el funtor N de la N-categoría.

6.1.3. ESTRUCTURA DE "CONJUNTO DE GUARDAS".

Según las definiciones anteriores se puede ver que, efectivamente, la N-categoría cumple las propiedades **[F28]** de definición del conjunto de guardas :
dada una N-categoría cualquiera C, para todo objeto de ella, c, existe uno correspondiente Nc que cumple:

$$c + Nc = < [Nc , c] , N < Nc , c > >$$

(es decir, existe, pues en C existen productos y coproductos finitos); además, el valor de
$c + Nc = < [Nc , c] , N < Nc , c > > = < 1 , N0 >$ de acuerdo con 6.1.1.1. y 6.1.1.2 ;
y como $N0 = 1$ y $< 1 , 1 > = 1$ (ver 6.1.1.4) se tiene que $c + Nc = 1$.

Por otro lado, $c \cdot Nc = Nc \cdot c = < c , Nc > = < Nc, c > = 0$ según 6.1.1.2, con lo que se obtiene estructura de "conjunto de guardas" para cualquier N-categoría.

6.2. EL PREORDEN INDUCIDO POR LA SUMA COINCIDE CON LA FLECHA DE LA N-CATEGORIA.

La suma que define el conjunto G(X) (ver [F28]) permite inducir sobre tal conjunto un preorden estableciendo que:
$f \leq g$ si existe un h tal que $g = f + h$ para $f, g, h \in G(X)$.
Esta relación, $\leq$, coincide con la flecha (morfismos) de la N-categoría sobre la que se ha definido la suma (ver 6.1.2.1).

6.2.1 En efecto, dados dos objetos c_1 y c_2 de la N-categoría C, se dice que $c_1 \to c_2$ si y solo si $[Nc_1 , c_2] = 1$ (ver d) de [F11]). Pero sabemos que $[Nc_1 , c_2] = < [Nc_1 , c_2] , 1 >$ por 6.1.1.4,

y $< [N c_1 , c_2] , 1 > = < <1 , [Nc_1 , c_2] > , 1 > =$

$= < < [Nc_1 , NNc_1] , [Nc_1 , c_2] > , No >$ por 6.1.1.3 y

$< < [Nc_1 , NNc_1] , [Nc_1 , c_2] > , No > =$

$= < [Nc_1 , < NNc_1 , c_2 >] , N < Nc_1 , < c_1 , c_2 > > >$

por la propiedad distributiva de los productos y porque

0 = (por 6.1.1.2) = $< Nc_1 , c_1 >$ =

= $< Nc_1 , c_1 >< Nc_1 , c_2 > = < Nc_1, < c_1 , c_2 > >$ y como c_1 = NNc_1, en definitiva,

$$[Nc_1 , c_2] =$$

= $< [Nc_1 , < NNc_1 , c_2 >] , N < Nc_1 , < NNc_1 , c_2 > > >$;

esta última expresión es una suma tal como está definida en 6.1.2.1 y se puede escribir, por tanto,

$[Nc_1, c_2] = Nc_1 + < NNc_1 , c_2 > = Nc_1 + < c_1 , c_2 >$ **[F34]**

es decir, $Nc_1 + < c_1 , c_2 > = 1$.

Por otro lado, $< Nc_1 , < c_1 , c_2 > > = 0$ y se vé que el producto $< c_1 , c_2 >$ cumple las condiciones del elemento c_1 respecto de su complementario Nc_1. Así pués, de la existencia de la flecha $c_1 \rightarrow c_2$ deducimos que $c_1 = < c_1 , c_2 >$ que es uno de los modos de caracterizar la ordenación $c_1 \leq c_2$ en el conjunto de las guardas, según **[F32]**.

6.2.2. Por el contrario, si suponemos que dos elementos, c_1 y c_2 del conjunto C considerado como conjunto de guardas cumplen que $c_1 \leq c_2$, sucederá que $c_1 \cdot c_2 = c_1$, es decir $< c_1 , c_2 > = c_1$ de tal modo que en la expresión anterior (**[F34]**, equivalencia válida en cualquier N-categoría) se obtiene que

$[Nc_1 , c_2] = Nc_1 + < c_1 , c_2 > = Nc_1 + c_1 = 1$,

es decir, $c_1 \rightarrow c_2$ en la categoría, según d) de [F11].

6.3. EQUIVALENCIA DE LA DISYUNCION EXCLUSIVA Y LA SUMA DE MULTIFUNCIONES.

Con la coincidencia demostrada entre el preorden inducido por la suma definida en 6.1.2.1 en el conjunto de las guardas (una N-categoría cualquiera considerada como conjunto de guardas) y la propia flecha de la N-categoría, se cierra el círculo de la equivalencia estructural entre los conjuntos de guardas y las N-categorías.

La definición del preorden introducido en el conjunto de guardas (en el contexto de la semántica parcialmente aditiva) se basa en la definición de la suma de multifunciones y funciones parciales realizada por Manes & Arbib (Véase [MAN86], páginas 28 y 29). Demostrada la equivalencia estructural entre el conjunto de las guardas y la N-categoría, la correspondencia demostrada en los apartados anteriores entre la suma en N-categorías (disyunción exclusiva) y dicha suma de multifunciones permite trasladar al ámbito de la semántica parcialmente aditiva los resultados desarrollados en N-categorías.

7. CONCLUSIONES Y DESARROLLOS FUTUROS

Tal como comentamos en la presentación de esta memoria hemos realizado un análisis desde el punto de vista de las N-categorías, de los elementos básicos que intervienen en la definición de la Lógica de la Programación (en su entorno de la Semántica de Aserciones), de la estructuración de las pre y postcondiciones de un fragmento de programa anotado a partir del operador wp y de la estructura interna de las guardas (tanto en la Lógica de la Programación como en el entorno de la Semántica Denotacional) así como otras conclusiones que hemos ido reseñando en los capítulos anteriores. Hemos obtenido diversos resultados individuales en cada uno de los apartados indicados y alguna conclusión general del examen conjunto de dichos resultados.

En efecto, a partir de la definición del operador wp hemos analizado los conjuntos de pre y postcondiciones de los programas anotados, y hemos llegado a

a) demostrar que el conjunto de precondiciones (igual que el de postcondiciones) tiene estructura de N-categoría (un tipo especial de N-categoría designada como E_{WSR}, para un programa S y una postcondición R) con elemento terminal, precisamente, el wp(S,R);

b) demostrar que el operador wp actúa como un funtor entre las N-categorías de precondiciones y postcondiciones que surgen a lo largo de un programa examinado fragmentadamente en módulos de programa; y que este funtor hace corresponder los objetos terminales con objetos terminales, los iniciales con iniciales y los productos y coproductos de ambas categorías (en el caso de los coproductos solo corresponden exactamente si los procesos son de tipo determinista).

Respecto de los conjuntos de guardas de un módulo de programa, se ha establecido:

c) que el conjunto de guardas de un programa tiene estructura de N-categoría, lo que supone que se puede operar en él utilizando todas las propiedades y métodos disponibles para estas estructuras y, en general, en todo el ámbito de las categorías;

d) en sentido inverso, hemos demostrado que toda N-categoría tiene estructura de "conjunto de guardas", tal como este tipo de conjunto ha sido definido por Manes & Arbib [MAN86] en el entorno de su semántica parcialmente aditiva;

e) en particular, además, hemos establecido que el preorden inducido en el conjunto G(X) por la suma, coincide con la flecha (morfismos) en la N-categoría;

f) como conclusión de esta cadena de resultados obtenidos, se ha puesto de manifiesto la equivalencia operacional entre la disyunción exclusiva definible en la N-categoría y la suma de multifunciones, lo que supone adicionalmente una presentación alternativa a la suma definida por Manes & Arbib en las categorías parcialmente aditivas; esta definición alternativa aparece sorprendentemente sencilla y simplifica enormemente los planteamientos subsiguientes.

Por último, del desarrollo conjunto de estas investigaciones y su relación se puede colegir que disponemos de nuevos instrumentos para entender, profundizar y desarrollar los diferentes conceptos involucrados en ambos planteamientos

(Semántica de Aserciones y Semántica Denotacional) y que las estructuras que subyacen en las precondiciones de un módulo de programa considerado dentro de una aserción de un programa anotado, coinciden con las de las funciones de guarda que definen y estudian Manes & Arbib en su Semántica Parcialmente Aditiva.

A partir de los resultados establecidos con el trabajo descrito en esta memoria, entendemos que pueden continuarse dos líneas de desarrollos posteriores, a cual más sugestiva: por un lado sería interesante la aplicación práctica de los fundamentos teóricos deducidos, al análisis y estructuración de módulos complejos de programas.

Estos esquemas establecidos, adecuadamente estructurados, podrían llevar a toda una metodología de desarrollo de programas sumamente eficaz y que, intuimos, debería ser bastante sencilla de manejo y comprensión. La abstracción al marco general de la teoría de categorías de los problemas concretos de implementación de módulos de programa debería ofrecer una simplicidad y potencia de cálculo grande.

Por otro lado, la unificación parcial realizada entre los modelos de razonamiento de la Semántica Parcialmente Aditiva (Manes & Arbib) y de la Lógica de Programación basada en el operador wp (Dijkstra y, sobre todo, Gries) podría extenderse a otros esquemas de desarrollo de técnicas de validación de programas (sistemas axiomáticos diversos) para su examen conjunto en el marco de la teoría de categorías. Ello supondría un marco general de razonamiento y podría aportar, creemos, alguna herramienta conceptual útil ó, al menos, un punto de vista diferente y, por diferente, posiblemente fructífero.

8. BIBLIOGRAFIA

Se incluye a continuación la reseña de los principales documentos (libros y artículos) utilizados en la preparación y desarrollo del presente trabajo. Aparecen solamente aquellos documentos de los que se ha hecho una consulta más continuada ó cuyo contenido es básico en los desarrollos realizados, y no aquellos otros a los que se ha accedido en un momento concreto. Aún así, son documentos de diferente utilización:

a) hay libros de consulta y documentación general, bien sobre lógica en general (como los de Kleene [KLE52], Haack [HAA74] y [HAA78] ó Turner [TUR84]), sobre teoría de conjuntos y categorías (por ejemplo, Jech [JEC78], Mitchell [MIT65], Freyd [FRE64], etc), sobre lógica de la programación (como el trabajo de Cook [COO78], el libro de Constable [CON78] y otros utilizados parcialmente) ó, incluso, sobre programación en general (se incluye, por ejemplo, la reseña del libro de Jensen & Wirth [JEN74]);

b) por otro lado están los documentos en que se desarrollan ciertas teorías ó técnicas novedosas (en su momento) que han

dado origen a los desarrollos de base de los que este trabajo arranca, tanto en el área de la semántica denotacional y de la semántica de aserciones, como de la lógica de la programación: así, se reseñan los trabajos de Floyd [FLO67], Elgot [ELG75], Hoare [HOA69], Dijkstra (tanto el de estructuración de la programación de 1972 [DIJ72] como el de presentación del operador wp [DIJ77]), etc.;

c) se incluyen también los documentos más recientes donde se presentan las últimas aportaciones en el área de nuestro interés: los libros de Manes & Arbib [MAN86], Gries [GRS81], Kfoury [KFO82] y otros; también algunos de los trabajos más interesantes para nosotros presentados en la Conferencia "Category Theory and Computer Programming" celebrada en la Universidad de Surrey (Guildford), en septiembre de 1985;

d) así mismo, se reseñan otros trabajos de reciente aparición (alguno aún sin publicar) que abordan aspectos relacionados con los desarrollos que hemos realizado (los no publicados los hemos conocido por haber sido preparados por personas con las que hemos comentado el contenido de nuestro trabajo y/ó por los directores del mismo).

[ALA78] Alagic, S. & Arbib, M.A. "The Design of Well Structured & Correct Programs". 1978. Springer-Verlag.

[BUR86] Burstall, R. and Rydeheard, D. "Computing with Categories". Proceedings de la Conferencia de Guildford. 1986. Springer-Verlag.

[CON78] Constable, R.L. and O'Donnell, M. "A Programming Logic". 1978. Winthrop Publishers.

[COO78] Cook, S.A. "Soundness & Completeness of an Axiom System for Program Verification". 1978. SIAM Journal on computing n° 7 (1978). Pgss. 70-90.

[CUE85] Cuena, J. "Logica Informática". 1985. Alianza Editorial.

[DBK80] De Bakker, J. "Mathematical Theory of Program Correctness". 1980. Prentice-Hall.

[DIJ72] Dijkstra, E.W. "Notes on Structured Programming". en el libro de Dahl, O., Hoare, C.AR. y Dijkstra, E.W. "Structured Programming". 1972. Academic Press. Posteriormente se publicó también como un informe técnico independiente.

[DIJ77] Dijkstra, E.W. "A discipline of Programming". 1977. Prentice-Hall.

[DYB86] Dybjer, P. "Category Theory and Programming Language Semantics: an Overview". En los Proceedings de la Conferencia de Gildford (Septiembre 1985). Springer. 1986.

[DOM89] Dominguez, E. y Lapeña, M.J. "Una Aproximación Lógica a la Verificación Automática de Programas Estructurados". 1989. (de pronta publicación)

[ELG75] Elgot, C.C. Lectures at the Stevens Institute of Technology. 1975.

[FLO67] Floyd, R. "Assigning Meaning to Programs". 1967. Mathematical Aspects of Computer Science. XIX American Mathematical Society, 19-32.

[FRE64] Freyd, P. "Abelian Categories". 1964. Harper & Row.

[GRS81] Gries, D. "The Science of Programming". 1981. Springer-Verlag.

[HAA74] Haak, S. "Deviant Logic". 1974. Cambridge University Press.

[HAA78] Haak, S. "Philosophy of Logics". 1978. Cambridge University Press.

[HAL62] Halmos, P.R. "Algebraic Logic". 1986. Chelsea Publishing Company.

[HOA69] Hoare, C.Ar. "An Axiomatic Approach to Computer Programming". 1969. Comm. ACM 12 (oct 69), 576-580.

[JEC78] Jech, T. "Set Theory". 1978. Academic Press.

[JEN74] Jensen, K. & Wirth, N. "Pascal. Users Manual & Report". 1974. Springer-Verlag.

[KFO82] Kfoury, A.J. , Moll, R.M. & Arbib, M.A. "A Programming Aproach to Computability". 1982. Springer-Verlag.

[KLE52] Kleene, S. "Introduction to Metamathematics". Van Nostrand.

[LAI74] Laita, L.M. "Un Estudio de la Lógica Algebraica desde el Punto de Vista de la Teoría de Categorías". 1974. Notre Dame Journal of Formal Logic Vol. XVII. n° 1. Jan 1976.

[LAI87] Laita, L.M. y Riscos, A. "N-categories in Logic". 1987. Zeitschr. f. Math. Logik und Grundlagen d. Math. Bd. 33, S. 507-516.

[LEB82] Leblanc, L. "Introduction à la Logique Algébrique". 1982. Département Mathem. Université de Montréal.

[LED88] Ledesma, L. de & Laita, L.M. "A Logic of Guards and Guarded Functions". Universidad de Praga. Report de junio 1988.

[LED89a] Ledesma, L. de & Laita, L.M. "An Aproach to Semantics of Programs from Preorder Categories". 1989. (de pronta publicación).

[LED89b] Ledesma, L. de. "Formas de Razonamiento".1989. (Comunicación privada).

[MCL72] Mac Lane, S. "Categories for the Working Mathematician". 1972. Springer-Verlag.

[MAN86a] Manes, E.G. & Arbib, M. A. "Algebraic Approaches to program semantics".1986. Springer-Verlag.

[MAN86b] Manes, E.G. "Weakest preconditions: categorical insights" Proceedings de la Conferencia de Guildford. 1986. Springer-Verlag.

[MIT65] Mitchell, B. "Theory of Categories". 1965. Academic Press.

[RYD88] Rydeheard, D.E. & Burstall, R.M. "Computational category theory". 1988. Prentice Hall.

[SCO71] Scott, D.S. & Strachey, C. "Towards a Mathematical Semantics for Programming Languages". en los "Proc. Symp. on Computers and Automata", editados por Fox, J. en 1971, pgs. 19-46. Polytechnic Institute of Brooklyn Press.

[STO79] Stoy, J. "Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory". 1979. M.I.T. Press.

[TUR84] Turner, R. "Logics of Artificial Intelligence". 1984. Ellis Horwood Ltd.

[WIR71] Wirth, N. "Program Development by Stepwise Refinement". Comm. A.C.M. (April 1971), Pgs 221-227.